БЕЛКООПСОЮЗ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БЕЛОРУССКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ
УНИВЕРСИТЕТ ПОТРЕБИТЕЛЬСКОЙ КООПЕРАЦИИ»

Кафедра информационно-вычислительных систем

СИСТЕМЫ БАЗ ДАННЫХ

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ И РАБОТА С НИМИ В СРЕДЕ СУБД MS ACCESS

Пособие
для студентов специальности 1-26 03 01
«Управление информационными ресурсами»

Гомель 2010

УДК 004.42
ББК 32.973.26-018.2
С 40

Авторы-составители: С. М. Мовшович, канд. техн. наук, доцент;
К. Г. Сулейманов, канд. техн. наук, доцент

Рецензенты: Е. В. Коробейникова, ст. преподаватель кафедры информационных технологий Гомельского государственного технического университета им. П. О. Сухого;
А. Н. Семенюта, д-р техн. наук, доцент, профессор кафедры информационно-вычислительных систем Белорусского торгово-экономического университета потребительской кооперации

Рекомендовано к изданию научно-методическим советом учреждения образования «Белорусский торгово-экономический университет потребительской кооперации». Протокол № 1 от 13 октября 2009 г.

С 40 **Системы** баз данных. Реляционные базы данных и работа с ними в среде СУБД MS Access : пособие для студентов специальности 1-26 03 01 «Управление информационными ресурсами» / авт.-сост. : С. М. Мовшович, К. Г. Сулейманов. – Гомель : учреждение образования «Белорусский торгово-экономический университет потребительской кооперации», 2010. – 120 с.
ISBN 978-985-461-736-7

УДК 004.42
ББК 32.973.26-018.2

ISBN 978-985-461-736-7

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Настоящее пособие предназначено для студентов специальности «Управление информационными ресурсами», изучающих дисциплину «Системы баз данных» («Введение в системы баз данных») и содержит теоретический материал, необходимый для подготовки студентов к экзамену и выполнению курсовой работы. В пособие входит лабораторный практикум, предполагающий физическое проектирование и использование базы данных. Кроме того, пособие содержит все необходимые сведения для подготовки студентов заочной формы обучения к компьютерному тестированию.

Пособие состоит из шести разделов. В первом разделе приводятся основные понятия реляционной базы данных на примере небольшой реляционной базы данных *Поставщики-Детали*. Здесь же приводятся определения основных операций реляционной алгебры Кодда: проекция, выборка, экви-соединение, естественное соединение и декартово произведение. Кроме того дается определение левого внешнего и правого внешнего соединений, которые непосредственно поддерживаются в СУБД Access. Все указанные операции важны для понимания того, каким образом реализуются запросы в СУБД Access с использованием *Конструктора запросов*.

Во втором разделе описываются основные понятия *ER-метода* логического проектирования реляционной базы данных (РБД), широко используемого на практике [1, с. 531–553], [4, с. 4–29].

Третий раздел представляет собой небольшой лабораторный практикум, предназначенный для выработки у студентов практических навыков по созданию реляционной базы данных и работе с ней в среде СУБД Access. Лабораторный практикум рассчитан на четыре часа аудиторных занятий.

В четвертом разделе уточняется понятие межтабличных связей в СУБД Access и представлено обобщенное описание формирования запросов с использованием *Конструктора запросов* СУБД Access.

В пятом разделе пособия приводится краткое описание языка *SQL*: дается общее представление о его специфике и возможностях, что позволяет начать самостоятельную работу в среде выбранной СУБД; краткие сведения о типах данных *SQL* и о двух его основных подмножествах *DDL* и *DML*.

В шестом разделе рассматриваются методы нормализации баз данных. Нормализация позволяет устранить излишнюю избыточность и аномалию обновления отношений баз данных (БД), которые особенно актуальны для тех реляционных БД, которые используются в информационных системах оперативной обработки транзакций.

1. РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

Реляционные базы данных в настоящее время наиболее распространены и фактически являются промышленным стандартом. Реляционная модель баз данных, базирующаяся на математической *теории отношений*, была предложена американским ученым Е. Ф. Коддом в начале 70-х гг. XX в.

1.1. Структура реляционных данных

Важнейшие понятия, используемые при описании реляционных баз данных, – домен, кортеж и отношение.

Эти понятия можно пояснить на конкретном примере представления в реляционной базе данных сущности реального мира «Деталь».

Домен представляет собой множество элементов, обладающих некими общими свойствами. Общие свойства элементов домена проявляются в том, что эти элементы в компьютере представляются данными одного типа.

В данном примере используются четыре домена с названиями *Коды деталей*, *Названия деталей*, *Цвета деталей* и *Веса деталей*.

Домен *Коды деталей* содержит строковые элементы с кодами деталей вида «д1», «д2», …, «дn», домен *Названия деталей* содержит строковые элементы со всевозможными названиями деталей, домен *Цвета деталей* содержит строковые элементы со всевозможными цветами деталей, домен *Веса деталей* содержит числовые элементы, соответствующие всевозможным весам деталей в определенной единице измерения веса.

Предполагается, что домены пронумерованы. В нашем примере будем считать, что первый домен – это домен *Коды деталей*, второй – *Названия деталей*, третий – *Цвета деталей*, четвертый – *Веса деталей*. При необходимости домены можно перенумеровать.

Кортеж представляет собой последовательность таких элементов, при которых первый элемент берется (копируется) из первого домена, второй – из второго домена и т. д. В приведенном примере кортеж будет состоять из четырех элементов из четырех доменов.

Отношение представляет собой множество однотипных кортежей. Это означает, что все кортежи отношения формируются по одной схеме: их первые элементы берутся (копируются) из первого домена, вторые – из второго домена и т. д.

Все вышесказанное о доменах, кортежах и их отношении проиллюстрировано на рис. 1, на котором домен *Коды деталей* обозначен как D1, *Названия деталей* – D2, *Цвета деталей* – D3 и домен *Веса деталей* – D4.

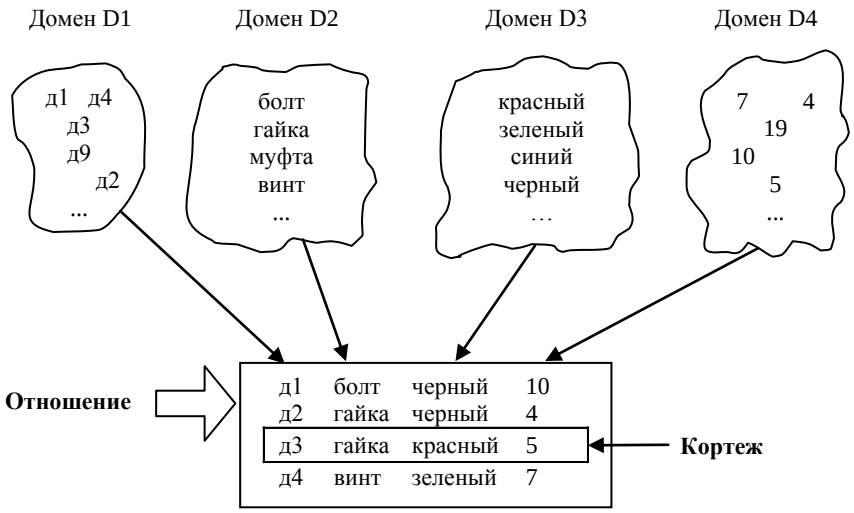


Рис. 1. **Отношение с математической точки зрения**

Домены D1, D2 и D3 – это множества символьных строк, при этом элементы этих множеств различаются количеством символов в них и (или) своей семантикой, т. е. смыслом, а D4 – множество целых чисел.

Таким образом, элементы домена обладают некими общими свойствами. Кроме того, предполагается, что над элементами одного домена определены допустимые для них операции. Например, для числового домена определены арифметические операции, для домена, содержащего строковые элементы, предусмотрена операция конкатенации (сцепления) и т. д. Предполагается также, что элементы одного домена могут сравниваться на равенство.

Примечание. Известный специалист по базам данных К. Дейт в своей книге «Введение в систему баз данных» [1] предпочитает использовать термин *тип* данных вместо *домен*.

Упорядоченность кортежей означает, что элементы каждого кортежа могут быть пронумерованы натуральными числами – 1, 2 и т. д.

Отношение состоит из четырех кортежей. Каждый кортеж – из четырех элементов, которые выбираются каждый из своего домена, т. е. первый элемент каждого кортежа выбран из домена D1, второй – из домена D2 и т. д.

Примечание. Не следует путать число кортежей отношения с количеством доменов, элементы которых используются в кортежах. Обычно число кортежей гораздо больше числа доменов. То, что их число в данном примере совпадает и равно четырем – чисто случайно.

Отметим также, что один и тот же элемент домена может повторяться в различных кортежах. Так, элемент «гайка» из домена D2 имеется во втором и третьем кортежах, а элемент «черный» из домена D3 – в первом и втором кортежах. Кроме того, не все элементы доменов могут быть использованы в отношении.

С точки зрения обработки данных отношение представляет собой простую двумерную таблицу (рис. 2). Поэтому на практике вместо термина *отношение* часто применяют термин *таблица*, вместо *кортеж* – *строка* или *запись*.

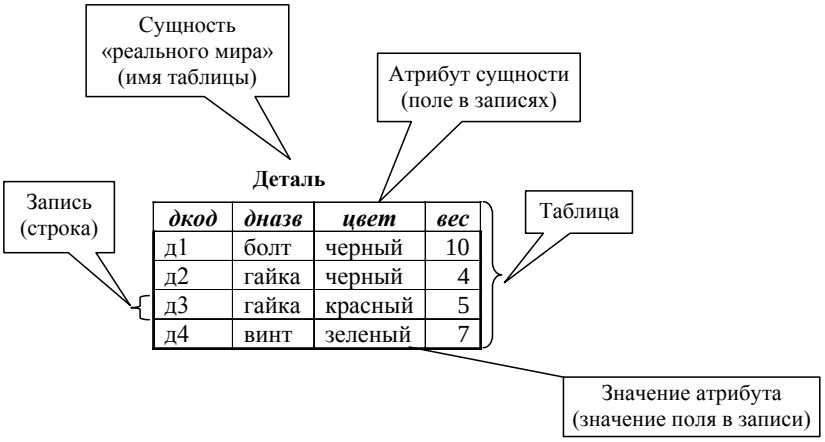


Рис. 2. **Отношение с точки зрения обработки данных**

Отношение соотносится с сущностью реального мира *Деталь*, а кортежи – с экземплярами этой сущности, т. е. с конкретной деталью. Таким образом, с точки зрения обработки данных отношение представляет собой простую двумерную таблицу, содержащую информацию об экземплярах некоторой сущности реального мира. Например, отношение на рис. 2 содержит информацию о четырех экземплярах сущности *Деталь*. Предполагается, что экземпляры сущности отличаются друг от друга, т. е. различимы. Поэтому в отношении не должно быть двух одинаковых кортежей.

Четыре домена, изображенных на рис. 1, соотносятся с четырьмя элементами реального мира: кодом детали, ее названием, цветом и весом.

Имена столбцов называются *атрибутами*, а индивидуальные значения, появляющиеся в отдельных кортежах, – *значениями атрибутов*.

Атрибуты могут располагаться в любом порядке, но от их переупорядочения смысл отношения не меняется. Атрибутам могут присваиваться произвольные названия. Однако необходимо учитывать, что в одном отношении не может быть атрибутов с одинаковыми названиями.

Кроме того, если атрибуты, а точнее было бы сказать значения атрибутов, двух отношений принадлежат одному домену, то им могут присваиваться как одинаковые, так и различные имена. В приводимых далее примерах таким атрибутам будут даваться одинаковые имена. В базе данных *Поставщики-Детали*, о которой будет говориться далее, такими атрибутами являются атрибуты *пфам* из отношений *Поставщик* и *Поставки*, а также *дкод* из отношений *Деталь* и *Поставки*.

Все случаи, когда атрибуты из разных отношений берутся из одного домена и имеют разные имена, будут каждый раз специально оговариваться.

На практике вместо термина *атрибут* могут использоваться термины *столбец* или *поле* (соответственно, вместо *значение атрибута* в кортеже – *значение столбца* или *значение поля* в записи).

Так как у множества элементы различимы, т. е. отличаются друг от друга, то у отношения нет двух одинаковых кортежей, так как отношение – это множество, элементами которого являются кортежи. На практике в таблице могут встречаться две идентичные записи – это одно из отличий отношений от таблиц, которые встречаются в бумажных документах.

Примечание. Еще одно отличие отношений от таблиц заключается в том, что при необходимости в отношениях кортежи неупорядочены. В таблице же строки могут быть упорядочены (отсортированы).

Степень отношения – число атрибутов (столбцов) отношения.

Мощность отношения – текущее число кортежей отношения.

Степень отношения обычно не изменяется после создания отношения, но его мощность будет колебаться по мере добавления новых и удаления старых кортежей.

У отношения *Деталь* степень равна четырем. Его мощность в настоящий момент также равна четырем.

1.2. Реляционная база данных и ее свойства

Реляционная база данных – совокупность отношений, содержащих всю информацию, которая должна храниться в БД.

В одной базе данных не должно быть отношений с одинаковыми именами.

На рис. 3 приведен пример очень маленькой БД, названной *Поставщики-Детали*.

Деталь : таблица

дкод	дназв	цвет	вес
д1	болт	черный	10
д2	гайка	черный	4
д3	гайка	красный	5
д4	винт	зеленый	7

Поставщик : таблица

пфам	город
Смит	Лондон
Джонс	Париж
Грис	Лондон

Поставки : таблица

пфам	дкод	шт
Джонс	д1	9
Джонс	д2	4
Смит	д1	3
Смит	д2	7
Смит	д3	2

Рис. 3. **База данных *Поставщики-Детали***

База данных *Поставщики-Детали* содержит три класса информации о некоторой компании.

Информация *первого класса* – информация о поставщиках, поставляющих детали предприятию. К такой информации относят фамилию поставщика (предполагается уникальной) и город проживания, не являющийся уникальным, т. е. в одном городе может быть несколько поставщиков с разными фамилиями. Эта информация содержится в отношении *Поставщик*, степень которого равна двум, а мощность – трем.

Информация *второго класса* – информация о деталях, используемых на предприятии. К ней относят код детали, являющийся уникальным, название, цвет и вес, не являющиеся уникальными. Такая информация содержится в отношении *Деталь*.

Информация *третьего класса* – информация о кодах и количестве деталей от каждого поставщика. Эта информация содержится в отношении *Поставки*, степень которого равна трем, а мощность – пяти.

Структура таблицы, используемой для хранения отношения, довольно проста, поскольку все записи имеют одинаковый формат.

Обычно предполагается, что в реляционной базе данных значения атрибутов кортежей являются атомарными (неделимыми), а не множественными. Если отношение удовлетворяет условию атомарности значений атрибутов, то говорят, что отношение находится в первой нормальной форме (1НФ).

В первой нормальной форме отношения значения всех атрибутов отношения являются атомарными (неделимыми).

Отметим, что таблица с неатомарными значениями в ячейках легко приводится к требуемому виду путем тривиального дублирования значений атомарных полей. Так, таблица *Поставки1* (табл. 1), содержащая неатомарные значения, приводится в 1НФ путем дублирования значений атомарного поля *пфам* (таблица *Поставки* из БД *Поставщики-Детали*).

Таблица 1. **Поставки1**

пфам	*дкод*	*шт*
Джонс	д1	9
	д2	4
Смит	д1	3
	д2	7
	д3	2

Число отношений в БД и конкретные атрибуты, приписываемые каждому отношению, определяются в процессе проектирования. После стадии проектирования БД начинается процесс ее реализации на компьютере с помощью системы управления базами данных (СУБД).

Система управления базами данных – программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных в актуальном состоянии, а также осуществлять к ней контролируемый доступ.

В настоящее время разработано много СУБД, предназначенных для работы с реляционными базами данных. В данном пособии изучаются методы работы с реляционной БД в среде СУБД MS Access.

Отметим, что в среде (справочной системе) СУБД MS Access используется термин таблица вместо термина отношение.

Важнейшими понятиями, связанными с отношениями реляционной базы данных, являются понятия первичного и внешнего ключа отношения.

Первичный ключ – минимальный набор атрибутов (или один атрибут) отношения, который может быть использован *для однозначной идентификации* конкретного кортежа.

Зная, что отношение не может иметь два одинаковых кортежа, можно сделать вывод, что отношение всегда имеет первичный ключ (в крайнем случае, им может быть полный набор его атрибутов). Однако при определении первичного ключа должно соблюдаться требование «минимальности». Это значит, что если произвольный единичный атрибут исключить из первичного ключа, оставшихся атрибутов будет недостаточно для однозначной идентификации отдельных кортежей.

Примечание. Иногда бывает несколько вариантов для выбора первичного ключа. В этом случае говорят, что у отношения есть несколько *потенциальных ключей*. Один из потенциальных ключей назначается в качестве первичного ключа.

В отношении *Деталь* первичным ключом является атрибут *дкод*, т. е. в нем нет двух кортежей с одинаковым значением атрибута *дкод*.

В отношении *Поставщик* первичным ключом может быть атрибут *пфам*, так как в нем нет двух поставщиков с одинаковыми фамилиями.

Примечание. На практике могут встречаться одинаковые фамилии. В этом случае в отношение добавляется атрибут, значения которого могут использоваться как первичный ключ. Например, на предприятиях сотрудникам приписываются табельные номера с уникальными значениями.

В отношении *Поставки* первичным ключом является пара атрибутов <*пфам, дкод*>, так как ни один из трех его атрибутов по отдельности не может однозначно идентифицировать конкретный кортеж.

Простой первичный ключ – первичный ключ, состоящий из одного атрибута.

Составной первичный ключ – первичный ключ, состоящий из двух или более атрибутов.

В базе данных *Поставщики-Детали* для отношений *Поставщик* и *Деталь* используется простой первичный ключ, у отношения *Поставки* – составной первичный ключ (первичные ключи отношений обычно подчеркиваются).

С первичным ключом связано понятие *целостности сущности*.

При *целостности сущности* атрибуты, входящие в первичный ключ, не должны содержать отсутствующих значений. Иногда отсутствующие значения обозначаются специальным определителем – *Null*.

Данное свойство первичного ключа вытекает из его определения. Так как первичный ключ однозначно определяет кортежи отношения, то значения остальных атрибутов отношения *зависят* от значений первичного ключа. Если допустить присутствие определителя *Null* в любом атрибуте первичного ключа, то это означает, что не все его атрибуты нужны для уникальной идентификации кортежей, что противоречит определению первичного ключа.

Если один или несколько атрибутов имеются в двух отношениях, то это обычно указывает на определенную *связь* между кортежами этих отношений.

Примечание. В СУБД MS Access имеются связи двух видов: связи ссылочной целостности и связи-соединения. Сейчас мы определим *связи ссылочной целостности*. Далее, при описании запросов с использованием *Конструктора запросов*, будут рассмотрен второй вид связи – *связи-соединения*.

Внешний ключ – атрибут или набор атрибутов в отношении *A*, который является первичным ключом в отношении *B*.

Отношение *B* называется *главным отношением* (*главной таблицей*) относительно отношения *A* по указанному внешнему ключу.

Примечание. На практике встречаются ситуации, когда отношение *B* может совпадать с отношением *A*. В таких случаях отношение *A* называется *самоссылающимся*. Более подробно об этом изложено в литературных источниках [1] и [3].

В отношении *Поставки* атрибут *пфам* является внешним ключом, так как он является первичным ключом в отношении *Поставщик*. Отношение *Поставщик* является главным относительно отношения *Поставки* по его внешнему ключу *пфам*.

Точно так же атрибут *дкод* в отношении *Поставки* является внешним ключом, так как он является первичным ключом в отношении *Деталь*. Отношение *Деталь* является главным относительно отношения *Поставки* по его внешнему ключу *дкод*.

Внешний ключ обычно используется для установления между двумя таблицами так называемой связи *ссылочной целостности*.

Ссылочная целостность означает, что если в отношении имеется внешний ключ, то его значение должно соответствовать значению первичного ключа в каком-либо кортеже главного отношения.

Ссылочная целостность требует, чтобы в отношении не было таких значений атрибутов, входящих во внешний ключ, которых нет в первичном ключе главного отношения. Такая согласованность значений внешних и первичных ключей является своеобразным обручем, скрепляющим воедино всю базу данных.

Обеспечение целостности сущностей и ссылочной целостности необходимо для того, чтобы база данных обладала так называемой *реляционной целостностью*.

Иногда для указания атрибутов, из которых состоят кортежи отношения, используется следующая форма записи: после имени отношения в скобках указывают атрибуты отношения. Поэтому отношения базы данных *Поставщики-Детали* могут быть записаны следующим образом:

Деталь (<u>дкод</u>, дназв, цвет, вес);

Поставщик (<u>пфам</u>, город);

Поставки (<u>пфам</u>, <u>дкод</u>, шт).

Первичные ключи подчеркнуты.

Состав полей записей соответствующих таблиц в среде СУБД MS Access может быть представлен графически (рис. 4). Первичные ключи таблиц выделены жирным шрифтом.

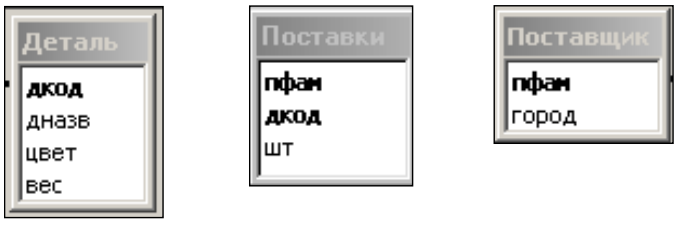


Рис. 4. **Графическое представление состава полей в записях таблиц в среде СУБД MS Access**

Ссылочная целостность таблицы *Поставки* по ее внешним ключам в среде СУБД MS Access представляется соответствующими линиями связи в схеме данных, как показано на рис. 5. Напомним, что у таблицы *Поставки* два внешних ключа. По внешнему ключу *дкод* она связана с таблицей *Деталь*, а по внешнему ключу *пфам* – с таблицей *Поставщик*.

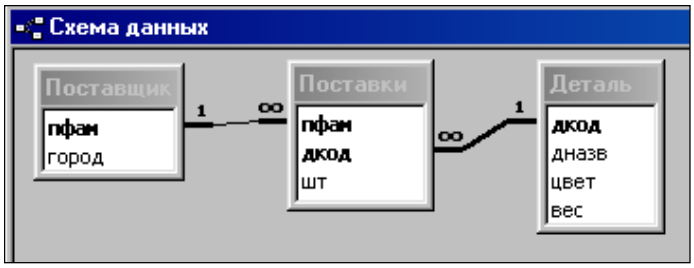


Рис. 5. **Схема данных БД *Поставщики-Детали* в среде СУБД MS Access**

В схеме данных линиями связываются внешний ключ одной таблицы и соответствующий первичный ключ другой (главной) таблицы. Символы «1» и «∞» на линиях связи указывают, что это связи типа «*один ко многим*». В данном случае это означает, что один поставщик может осуществить много поставок различных деталей и одна деталь может поставляться многократно различными поставщиками.

Примечания:

1. СУБД MS Access поддерживает также связи ссылочной целостности типа «*один к одному*». Такая связь устанавливается, когда у одной таблицы связываемое поле является первичным ключом, а у второй таблицы связываемое поле либо является первичным ключом, либо не является первичным ключом, но имеет только одно допустимое значение.

Подробнее об этом будет сказано в подразделе 2.7 «Правила генерации отношений по диаграммам *ER-типа*» в правилах 1–3.

2. Возможные в реальности связи типа «*многие ко многим*», когда одной записи таблицы A соответствуют несколько записей таблицы B и, наоборот, одной записи таблицы B соответствуют несколько записей таблицы A, сводятся к связям типа «*один ко многим*» таблиц A и B с третьей дополнительной (связующей) таблицей. В число полей третьей таблицы включаются поля первичных ключей таблиц A и B, которые, таким образом, становятся внешними ключами третьей таблицы.

К свойствам отношений и реляционной БД в целом можно отнести следующие:

- отношение имеет имя, которое отличается от имен всех других отношений рассматриваемой БД;
- любое значение каждого атрибута является атомарным;
- каждый атрибут в отношении имеет уникальное имя;
- все значения атрибута берутся из одного и того же домена;
- порядок следования атрибутов не имеет никакого значения;
- в одном отношении нет дубликатов кортежей;
- теоретически порядок следования кортежей в отношении не имеет никакого значения.

База данных должна обладать реляционной целостностью – целостностью каждой сущности и ссылочной целостностью.

1.3. Языки запросов к реляционным базам данных

Для того, чтобы пользователь мог выполнить с помощью СУБД вышеперечисленные действия с БД, обычно ему предоставляются языковые средства. В настоящее время существует много искусственных языков, предназначенных для описания пользователем необходимых ему действий по манипулированию данными в БД. Однако стандартом для реляционных БД является язык *SQL* (*Structured Query Language* – язык структурированных запросов), который в большей или меньшей степени поддерживает любая СУБД, претендующая на звание «реляционной».

Примечание. Существует несколько редакций стандарта языка *SQL*. Последняя редакция принята в качестве международного стандарта осенью 2003 г.

Необходимо отметить, что в названии языка *SQL* присутствует слово *Query* – *запрос*. Это связано с весьма важным обстоятельством, которому не во всех руководствах по работе с БД уделяется должное внимание. Среди всех многочисленных функций СУБД *главная – выдавать ответы на поступающие запросы*. Дело в том, что обычно базы данных содержат огромный объем информации, который пользователь не может в целом «переварить» и осмыслить. Он только может работать с отдельными «срезами» информации и извлекать их из БД с помощью запросов.

Хотя *SQL* и задумывался как средство работы конечного пользователя, в конце концов, он стал настолько сложным, что превратился в инструмент программиста, так объем опубликованного стандарта языка *SQL* более 2000 страниц.

Для пользователей, не являющихся профессиональными программистами, в СУБД Microsoft Access предусмотрен адаптированный язык запросов *QBE*, реализованный в виде *Конструктора запросов*. Именно *Конструктор* и будет использоваться при изучении запросов во втором и третьем разделах.

Важнейшими характеристиками реляционных языков является их селективная мощность и простота их изучения и использования.

Селективная мощность реляционных языков – относительная (сравнительная) характеристика языков, определяющая их возможности для получения требуемой информации из базы данных.

В качестве эталона при сравнении селективной мощности различных реляционных языков обычно принимается реляционная алгебра Кодда.

1.4. Операции реляционной алгебры

Реляционная алгебра Кодда содержит восемь операций: четыре теоретико-множественные операции (объединения, пересечения, разности и декартова произведения) и четыре специальные реляционные операции (выборки, проекции, соединения и деления). Схематическое изображение этих операций приведено на рис. 6.

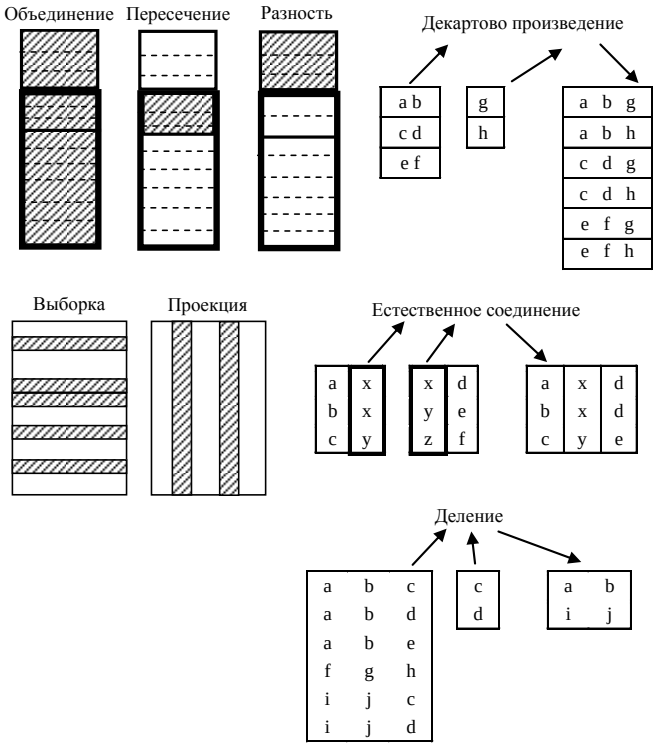


Рис. 6. **Операции реляционной алгебры**

Многие авторы считают, что язык является *реляционно полным*, если позволяет для любого конечного набора отношений *R*1, *R*2, …, *Rn* в первой нормальной форме определить любое отношение, выводимое из *R*1, *R*2, …, *Rn* с помощью выражений реляционной алгебры Кодда. Другими словами, выразительная мощность реляционно полного языка не должна уступать реляционной алгебре Кодда.

Операндами и результатами операций реляционной алгебры Кодда являются отношения. Операции выборки и проекции являются унарными, поскольку они работают с одним отношением. Другие операции работают с парами отношений, и поэтому их называют бинарными операциями.

В среде СУБД MS Access все восемь операций реляционной алгебры Кодда могут быть реализованы на языке *QBE*, т. е. с помощью *Конструктора запросов*. Другими словами, язык *QBE* является реляционно полным и его селективная мощность не меньше селективной мощности реляционной алгебры Кодда.

*Примечание***.** На самом деле селективная мощность языка *QBE* больше селективной мощности реляционной алгебры Кодда, так как в конструкторе запросов СУБД MS Access имеются дополнительные возможности (дополнительные операции) по работе с таблицами реляционной базы данных:

- возможность работы с частью значения атрибута (подстрокой атрибута-строки);
- возможность установления между таблицами левого и правого внешнего соединения;
- возможность создания перекрестного запроса, результатом которого является таблица, в которой на уровень названий атрибутов выносятся конкретные значения атрибутов, т. е. данные переносятся на уровень метаданных. Метаданные – это данные о данных.

Наиболее важными и часто используемыми при выполнении запросов к реляционной базе данных являются операции выборки, проекции, соединения и декартова произведения.

1.4.1. Декартово произведение (CARTESIAN PRODUCT)

Операцию обозначают следующим образом: $T = R \otimes S$.

Входные отношения *R* и *S* могут иметь разный состав атрибутов. Результирующее отношение *T* включает все атрибуты исходных отношений. Кортежами отношения *T* являются всевозможные попарные конкатенации («сцепления») кортежей отношений *R* и *S*. Таким образом можно сделать вывод, что степень

результирующего отношения *T* равна сумме степеней исходных отношений *R* и *S*, а мощность результирующего отношения *T* равна произведению мощностей исходных отношений *R* и *S*.

Пример 1. Пусть операндами будут отношения *Поставщик* и *Поставки*. Результатом операции *Поставщик* ⊗ *Поставки* является отношение (его степень 2 + 3 = 5, мощность 3 · 5 = 15), приведенное в табл. 2.

Таблица 2. **Декартово произведение отношений *Поставщик* и *Поставки***

Поставщик.пфам	*город*	*ПД.пфам*	*дкод*	*шт*
Смит	Лондон	Джонс	д1	9
Джонс	Париж	Джонс	д1	9
Грис	Лондон	Джонс	д1	9
Смит	Лондон	Джонс	д2	4
Джонс	Париж	Джонс	д2	4
Грис	Лондон	Джонс	д2	4
Смит	Лондон	Смит	д1	3
Джонс	Париж	Смит	д1	3
Грис	Лондон	Смит	д1	3
Смит	Лондон	Смит	д2	7
Джонс	Париж	Смит	д2	7
Грис	Лондон	Смит	д2	7
Смит	Лондон	Смит	д3	2
Джонс	Париж	Смит	д3	2
Грис	Лондон	Смит	д3	2

1.4.2. Выборка (SELECTION)

Операцию *Выборка* обозначают формулой $T = \sigma_C(R)$.

На входе используется одно отношение *R*, результат – новое отношение *T*, имеющее тот же состав атрибутов. Результирующее отношение *T* содержит подмножество кортежей исходного отношения *R*, удовлетворяющих условию выборки, которое обозначено буквой *C*.

Пример 2. Пусть требуется определить записи из таблицы *Деталь*, у которых значение поля *дназв* равно «гайка». Результатом операции $\sigma_{дназв}$ = «гайка» (*Деталь*) является отношение, приведенное в табл. 3.

Таблица 3. **Выборка из отношения *Деталь* записей с названием *гайка***

дкод	*дназв*	*цвет*	*вес*
д2	гайка	черный	4
д3	гайка	красный	5

1.4.3. Проекция (PROJECTION)

Обозначить операцию *Проекция* можно следующим образом: $T = \pi_{A1, A2, \ldots, An}(R)$.

Операция проекции производит выборку из каждого кортежа отношения-операнда *R* значений атрибутов *A*1, *A*2, …, *An*, входящих в список, и удаляет из полученной таблицы повторяющиеся строки.

Пример 3. Пусть требуется определить фамилии поставщиков, поставивших какие-либо детали. Ответ дает проекция отношения *Поставки* по атрибуту *пфам*.

В табл. 4 представлено отношение, являющееся результатом операции $\pi_{пфам}$ (*Поставки*). Отношение представлено в двух видах – предварительном и окончательном (после удаления повторяющихся кортежей).

Таблица 4. **Проекция отношения *Поставки* по атрибуту *пфам***

Предварительная таблица

пфам
Джонс
Джонс
Смит
Смит
Смит

Окончательная таблица

пфам
Джонс
Смит

В табл. 5 представлено отношение, являющееся результатом проекции отношения *Деталь* по атрибутам *дназв* и *цвет* $\pi_{дназв,\ цвет}$ (*Деталь*).

Таблица 5. **Проекция отношения *Деталь* по атрибутам *дназв* и *цвет***

дназв	*цвет*
болт	черный
гайка	черный
гайка	красный
винт	зеленый

1.4.4. Соединение (JOIN)

Операция соединение (*join*) имеет несколько модификаций: тета-соединение, экви-соединение, которое еще называют «соединение по эквивалентности», естественное соединение, левое внешнее соединение, правое внешнее соединение и полное внешнее соединение.

В СУБД Access могут устанавливаться связи-соединения, позволяющие непосредственно реализовать экви-соединение, естественное соединение, левое внешнее соединение и правое внешнее соединение.

Экви-соединение. Обозначение операции можно представить следующим образом: $T = R \rhd\lhd_C S$.

Отношения-операнды R и S могут иметь разный состав атрибутов. Однако в R должны быть атрибуты $A1, A2, \ldots, An$, а в S – атрибуты $B1, B2, \ldots, Bn$ такие, что $A1$ и $B1$ определены на домене $D1$, $A2$ и $B2$ – на домене $D2$, …, An и Bn определены на домене Dn. Таким образом, значения атрибутов из каждой пары могут быть сравнены на совпадение (равенство). Это условие сравнения на совпадение (равенство) обозначено буквой C.

Примечание. На самом деле условие C имеет следующий вид:

$$(A1 = B1)\ \&\ (A2 = B2)\ \&\ \ldots\ \&\ (An = Bn), \qquad (1)$$

где символом & обозначена операция логического умножения.

Во всех приводимых здесь примерах $n = 1$, т. е. соединение производится по одному полю.

Результирующее отношение T включает все атрибуты исходных отношений R и S. Кортежи результирующего отношения T являются конкатенацией кортежей исходных отношений R и S, для которых выполняется условие C, т. е. значение атрибута $A1$ из кортежа отношения R равно значению атрибута $B1$ из кортежа отношения S, значение атрибута $A2$ из кортежа отношения R равно значению атрибута $B2$ из кортежа отношения S, а значение атрибута An из кортежа отношения R равно значению атрибута Bn из кортежа отношения S.

Очевидно, в результирующем отношении T будет n пар одинаковых столбцов. Также как и в случае с декартовым произведением степень результирующего отношения T равна сумме степеней исходных отношений R и S.

Операцию экви-соединения можно переписать на основе операций выборки и декартова произведения. Она будет выглядеть следующим образом:

$$R \rhd\lhd_C S = \sigma_C(R \otimes S). \qquad (2)$$

Из этой формулы видно, что результат экви-соединения двух отношений представляет собой подмножество декартова произведения этих же отношений: из отношения, представляющего собой декартово произведение, выбираются только те кортежи, которые удовлетворяют условию C.

Примечание. Из сказанного не следует делать вывод о том, что можно обойтись без операции экви-соединения. Дело в том, что операция декартова произведения достаточно трудоемкая. Операция экви-соединения и была введена для того, чтобы избежать вычисления декартова произведения.

Естественное соединение (обозначение $T = R \rhd\lhd S$) аналогично экви-соединению, но есть несколько существенных отличий:

- предполагается, что атрибуты каждой пары Ai и Bi $(1 \le i \le n)$, значения которых проверяются на совпадение, должны иметь одинаковые имена. Это может быть сделано, например, путем переименования всех Bi на Ai $(1 \le i \le n)$ перед выполнением естественного соединения отношений R и S;
- в результирующем отношении T из каждой пары атрибутов Ai и Bi $(1 \le i \le n)$ остается только один атрибут.

Так как атрибуты Ai и Bi $(1 \le i \le n)$ имеют одинаковые имена, то не возникает вопросов, какой атрибут из R сравнивается на совпадение с каким атрибутом из S, и какое имя дать атрибуту, остающемуся в результирующем отношении T.

Степень результирующего отношения T естественного соединения равна сумме степеней отношений-операндов R и S минус количество общих атрибутов n.

Примечание. Естественное соединение отношений *R* и *S* можно получить из их экви-соединения, применив к последнему операцию проекции. Можно показать, что в данном случае результат операции проекция не содержит кортежей-дубликатов.

На практике из всех видов операций соединения чаще всего используется естественное соединение. *Поэтому, если иное специально не оговорено, то под операцией соединения обычно понимают операцию естественного соединения.*

Пример 4. Для отношений *Поставки* и *Поставщик* экви-соединение *Поставки* $\triangleright\triangleleft_{\text{Поставки.пфам=Поставщик.пфам}}$ *Поставщик* (по атрибуту *пфам*, являющемуся первичным ключом в отношении *Поставщик* и внешним ключом в отношении *Поставки*), представляет собой отношение со степенью равной 3 + 2 = 5, которое приведено в табл. 6.

Таблица 6. **Экви-соединение отношений *Поставки* и *Поставщик***

Поставки.пфам	*дкод*	*шт*	*Поставщик.пфам*	*город*
Джонс	д1	9	Джонс	Париж
Джонс	д2	4	Джонс	Париж
Смит	д1	3	Смит	Лондон
Смит	д2	7	Смит	Лондон
Смит	д3	2	Смит	Лондон

Видно, что у таблицы-результата два одинаковых столбца. Один столбец содержит значения поля *пфам* из таблицы *Поставки*, другой – из таблицы *Поставщик*. Этот факт обозначается приписыванием перед названием поля имени таблицы и точки.

Результатом естественного соединения этих же отношений *Поставки*$\triangleright\triangleleft$*Поставщик* (табл. 7) является отношение со степенью 3 + 2 – 1 = 4, содержащее такое же число записей, что и при экви-соединении (см. табл. 6), но содержит на один столбец меньше, чем при экви-соединении рассматриваемых отношений.

Таблица 7. **Естественное соединение отношений *Поставки* и *Поставщик***

пфам	*дкод*	*шт*	*город*
Джонс	д1	9	Париж
Джонс	д2	4	Париж
Смит	д1	3	Лондон
Смит	д2	7	Лондон
Смит	д3	2	Лондон

Иногда необходимо, чтобы запись из одной таблицы была представлена в таблице-результате соединения, даже если в другой таблице нет записи с совпадающим значением сравниваемого поля. Такой результат достигается с помощью внешнего соединения. Достоинством внешнего соединения является то, что при таком соединении сохраняется исходная информация, т. е. внешнее соединение сохраняет записи, которые были бы утрачены при других видах соединения.

Левым внешним соединением, обозначаемым как $T = R \supset\triangleleft S$, называется соединение, при котором записи таблицы *R*, не имеющие совпадающих значений в общих столбцах таблицы *S*, также включаются в результирующую таблицу *T*. Предполагается, что поля отношений *R* и *S*, по которым производится соединение, имеют одинаковые имена как и при естественном соединении.

Для обозначения отсутствующих значений во второй таблице *S* используется пустое (неопределенное) значение *Null*.

Пример 5. Левое внешнее соединение *Деталь* $\supset\triangleleft$ *Поставки* таблиц *Деталь* и *Поставки* по полю *дкод*, являющемуся первичным ключом в таблице *Деталь* и внешним ключом в таблице *Поставки*, представлено табл. 8.

Таблица 8. **Левое внешнее соединение отношений *Деталь* и *Поставки***

дкод	*дназв*	*цвет*	*вес*	*пфам*	*шт*
д1	болт	черный	10	Джонс	9
д1	болт	черный	10	Смит	3
д2	гайка	черный	4	Джонс	4
д2	гайка	черный	4	Смит	7
д3	гайка	красный	5	Смит	2
д4	винт	зеленый	7	*Null*	*Null*

Аналогично определяется *правое внешнее соединение*, обозначаемое как $T = R \triangleright\subset S$.

Пример 6. Правое внешнее соединение *Поставки* $\triangleright\subset$ *Поставщик* таблиц *Поставки* и *Поставщик* по полю *пфам*, являющемуся первичным ключом в таблице *Поставщик* и внешним ключом в *Поставки*, представлено табл. 9.

Таблица 9. **Правое внешнее соединение отношений *Деталь* и *Поставки***

дкод	*шт*	*пфам*	*город*
Null	*Null*	Грис	Лондон
д1	9	Джонс	Париж
д2	4	Джонс	Париж
д1	3	Смит	Лондон
д2	7	Смит	Лондон
д3	2	Смит	Лондон

Примечание. Внешние соединения (левое, правое и полное) кардинально отличаются от других видов соединения, так как результат внешнего соединения двух отношений не является подмножеством декартова произведения этих же отношений, а в результирующем отношении имеются неопределенные значения *Null*.

Многие специалисты, в частности Дейт [1], считают недопустимым появление в отношениях неопределенного значения *Null*, так как это нарушает логическую стройность теории реляционных баз данных из-за того, что во многих случаях двузначную логику (с двумя логическими значениями *ЛОЖЬ* и *ИСТИНА*) приходится заменять трехзначной логикой (добавляется третье логическое значение *НЕИЗВЕСТНО*).

Тем не менее, левое (*LEFT JOIN*), правое (*RIGHT JOIN*) и полное (*FULL JOIN*) внешние соединения включены в стандарт языка *SQL* [2]. В СУБД MS Access также реализованы левое и правое внешние соединения.

На примере, представленном в табл. 10, приведем сводку рассмотренных разновидностей операции соединения.

Таблица 10. **Отношения *R* и *S***

R		*S*	
A	B	C	D
x	1	1	i
y	3	5	j
z	5	9	k

Пусть даны таблица *R* с атрибутами A, B и таблица *S* с атрибутами C, D.

Будем считать, что значения атрибутов *B* и *C* принадлежат одному домену целых чисел, поэтому могут сравниваться.

Результат экви-соединения $R\triangleright\triangleleft_{B=C}S$ отношений *R* и *S* представлен в табл. 11.

Таблица 11. **Экви-соединение отношений *R* и *S***

A	*B*	*C*	*D*
x	1	1	i
z	5	5	j

Результат естественного соединения $R\triangleright\triangleleft S1$, где таблица *S*1 отличается от таблицы *S* только тем, что столбец (поле) C переименован в B, представлен табл. 12.

Таблица 12. **Естественное соединение отношений *R* и *S*1**

A	*B*	*D*
x	1	i
z	5	j

Результат левого внешнего соединения $R\supset\triangleleft S1$ представлен в табл. 13.

Таблица 13. **Левое внешнее соединение отношений *R* и *S*1**

A	*B*	*D*
x	1	i
y	3	*Null*
z	5	j

Результат правого внешнего соединения $R\triangleright\subset S1$ представлен в табл. 14.

Таблица 14. **Правое внешнее соединение отношений *R* и *S*1**

A	*B*	*D*
x	1	i
z	5	j
Null	9	k

Необходимо отметить, что из восьми операций реляционной алгебры только пять являются примитивами – выборка, проекция, декартово произведение, объединение и разность. Остальные три операции – пересечение, соединение и деление – могут быть определены через первые пять. Соответствующие формулы (аналогичные формуле 2) приводятся во многих литературных источниках по реляционным базам данных [2, с. 130, 131].

2. МЕТОД «*СУЩНОСТЬ – СВЯЗЬ*» ЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ (*ER-МЕТОД*)

2.1. Основные этапы проектирования базы данных

Создание базы данных в среде системы управления базами данных Access (СУБД Access) предполагает выполнение следующих основных этапов:

- концептуальное проектирование;
- логическое проектирование;
- физическое проектирование;
- использование БД (заполнение БД оперативной информацией и формирование запросов и отчетов).

Концептуальное проектирование – процедура конструирования информационной модели предприятия, не зависящей от условий реализации БД. Таким образом, сконструированная на данном этапе информационная модель не зависит ни от СУБД, ни от средств вычислительной техники.

Концептуальное проектирование БД выполняется на основе анализа информационных потоков организации, использования классификаторов и систем кодирования, определения диапазона действия и области применения БД, выяснения состава ее пользователей, сбора и анализа требований пользователей.

В настоящем пособии не рассматривается методика проведения концептуального проектирования. Предполагается, что к моменту начала логического проектирования базы данных сконструирована информационная модель рассматриваемой предметной области.

На этапе логического проектирования информационная модель предприятия уточняется с учетом типа создаваемой БД (реляционной, сетевой или иерархической). В настоящее время реляционные модели базы данных практически повсеместно вытеснили все другие типы моделей. В СУБД Access реализована именно реляционная база данных.

Процесс физического проектирования БД предполагает выполнение в среде выбранной СУБД следующих работ:

- описание логической структуры каждой таблицы;
- описание связей между таблицами, входящими в одну БД;
- первоначальное заполнение справочников БД необходимой нормативно-справочной информацией.

Подробные сведения об этапах жизненного цикла БД изложены в работах К. Дж. Дейта, Т. Коннолли и др.

Подчеркнем, что концептуальное проектирование БД не связано с какой-либо конкретной СУБД, а этап логического проектирования зависит только от типа СУБД (сетевой, иерархической или реляционной). Однако способ представления результатов концептуального проектирования зависит от используемого метода логического проектирования. Используемые термины «отношение» и «атрибут» относятся к реляционной СУБД, каковой и является СУБД Access.

Поскольку одна из целей нашего курса состоит в изучении технологии применения СУБД Access для реализации реляционных баз данных, то последние два этапа создания баз данных предполагают знание пользователем именно этой СУБД.

2.2. Концепция ER-метода логического проектирования

Существуют различные подходы представления информационной модели рассматриваемой предметной области как результата концептуального проектирования и, соответственно, как набора исходных данных для логического проектирования. В настоящем пособии рассматривается *ER-метод* логического проектирования.

Данный метод предполагает, что в результате выполнения концептуального проектирования БД должно быть достигнуто следующее:

- выделены все сущности, информация о которых должна содержаться в искомой БД;
- определены основные атрибуты для каждой сущности;
- назначен ключевой атрибут для каждой сущности;
- сформулированы связи между выделенными сущностями;
- выявлены условия применения выделенных сущностей на данном предприятии.

Таким образом, информационная модель предметной области должна включать указанные пять компонентов.

В качестве определения *ER-метода* можно принять следующее:

- основу *ER-метода* составляют понятия «сущность», «связь» и «атрибуты»;
- суть *ER-метода* состоит в наборе формализованных процедур и правил, позволяющих получить полный набор таблиц и определить структуру каждой таблицы на основе приведенных пяти компонентов информационной модели предметной области.

Особенностью *ER-метода* является то, что для одной и той же проблемы разные проектировщики могут получать различные наборы сущностей и связей. Определение лучшего из этих наборов может быть вопросом личного предпочтения. Само проектирование БД не является полностью формализуемым процессом, но *ER-метод* привносит в процесс проектирования достаточно много четких процедур.

2.3. Основные понятия

Сущность определяется как некоторый объект, представляющий интерес для предприятия, информация о котором должна храниться в БД. Этот объект должен иметь *экземпляры*, отличающиеся друг от друга и допускающие однозначную идентификацию. Сущность представляется, как правило, именем существительным. Примерами сущностей могут служить машины, банковские счета, учебные заведения, служащие и договоры.

Кроме выделения сущностей на этапе концептуального проектирования обычно определяют обязательный минимум свойств (атрибутов) каждого объекта.

Атрибут есть свойство сущности.

Например, атрибутами (свойствами) сущности *Студент* являются *Фамилия, Имя, Отчество, Номер зачетной книжки, Год рождения, Академическая группа* и т. д.

Среди множества атрибутов каждого объекта (сущности) должен присутствовать так называемый ключевой атрибут или ключ сущности, однозначно идентифицирующий конкретный экземпляр данной сущности.

Атрибут, или набор атрибутов, используемый для однозначной идентификации экземпляра сущности, называется *ключом сущности*.

Для сущности *Студент* ключом сущности является номер зачетной книжки. Определенное значение этого номера однозначно указывает на конкретного студента. Заметим, что совокупность атрибутов, определяющих фамилию, имя и отчество студента не является ключом сущности, так как в одном вузе могут учиться два студента с одинаковыми данными.

На этапе логического проектирования определяются все таблицы (*отношения*) БД и полный список их *атрибутов*. В некоторых книгах для таблиц (отношений), получаемых на этапе логического проектирования, используется термин «*информационный объект*».

Связь представляет собой соединение двух или более сущностей. При поиске связей в основном следует полагаться на то обстоятельство, что связь обычно выражается глаголом. Типичными примерами связей между двумя сущностями являются следующие: служащие *Работают* в отделах, студенты *Изучают* учебные предметы, рабочие *Обслуживают* механизмы (или механизмы *Обслуживаются* рабочими).

Характеристики связи во многом определяются условиями применения сущностей. *Условия применения* – это производственные правила, установленные в данной организации, использования выделенных для БД объектов.

2.4. ER-диаграммы

Методику построения *ER-диаграмм* рассмотрим на следующем примере. Предположим, что проектируется БД, предназначенная для хранения информации о распределении по складам (или по секциям склада) номенклатуры товарных продуктов, выпускаемых предприятием. В результате концептуального проектирования было выявлено, что двумя главными объектами, или *сущностями*, представляющими в данном случае интерес, являются *Продукт* и *Склад*.

Сущность *Продукт* характеризуется такими атрибутами, как *Номер продукта* (*НП*), *Наименование продукта* (*НАИМ*), *Единица измерения* (*ЕИ*), *Упаковка* (*УП*) и др. Для дальнейшего рассмотрения важно лишь то, что атрибут *НП* является ключом сущности *Продукт*. Это означает, что значение атрибута *НП* однозначно определяет конкретный продукт, т. е. *экземпляр сущности Продукт*. Поэтому будем считать, что *НП* принимает следующие значения: *П*1, *П*2, *П*3 и т. д.

Сущность *Склад* обладает следующими атрибутами: *Номер склада* (*НС*), *Емкость склада* (*ЕС*), *Материально ответственное лицо* (*МОЛ*) и т. д. Ключом сущности является атрибут *НС*. Будем считать, что *НС* принимает значения *С*1, *С*2, *С*3 и т. д.

Сущности *Продукт* и *Склад* соотносятся с помощью связи *Хранится*. Эта связь может быть графически представлена в виде диаграммы *ER-экземпляров* и диаграммы *ER-типа*.

Рис. 7 иллюстрирует использование диаграммы *ER-экземпляров* для примера, показывающего, на каком именно складе хранится каждый продукт. В этом примере каждый продукт (каждый экземпляр сущности *Продукт*) идентифицируется номером продукта (*НП*), а каждый экземпляр сущности *Склад* – номером склада (*НС*). Экземпляры сущностей изображаются на диаграмме точками, рядом с которыми написаны их идентификаторы (номера). Каждая линия связывает некоторую пару экземпляров сущностей (в данном случае *Продукт* и *Склад*) и показывает, что данная пара ассоциирована рассматриваемой связью (на каком конкретном складе находится данный продукт).

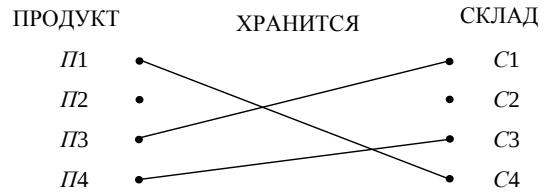


Рис. 7. **Пример диаграммы *ER*-экземпляров**

На диаграмме *ER-экземпляров*, представленной на рис. 7, названия всех сущностей (*Продукт, Склад*) помещаются над экземплярами этих сущностей, в то время как каждый экземпляр сущности идентифицируется значениями своих атрибутов или значением ключевого атрибута (*Номер продукта, Номер склада*). Так, *Продукт* является сущностью, а *П*1 – конкретным экземпляром сущности со значением ключевого атрибута *П*1. Связь также именуется, и ее название (*Хранится*) размещается над экземплярами связи, при этом экземпляр каждой отдельной связи специфицируется линией между теми двумя экземплярами сущностей, которые эта связь соединяет. Экземпляр связи между *С*1 и *П3*, например, означает, что продукт с номером продукта, равным *П3*, хранится на складе с номером склада *С*1.

Диаграмма, изображенная на рис. 8, называется *диаграммой ER-типа* и содержит ту же общую информацию, которая содержится на рис. 7. В то же время информации, представленной на рис. 8, достаточно для определения отношений БД.

Рис. 8. **Пример диаграммы *ER-типа***

На диаграммах ER-типа, подобных показанной на рис. 8, сущности представляются в виде прямоугольников, а связи – в виде ромбов. Непосредственно под прямоугольным блоком каждой сущности выписывается и выделяется подчеркиванием ключ этой сущности: *НП* (*Номер продукта*) для сущности *Продукт* и *НС* (*Номер склада*) для сущности *Склад*. Многоточие, расположенное вслед за каждым из этих атрибутов, указывает на то, что никакие другие возможно имеющиеся атрибуты соответствующей сущности не могут быть частью ее ключа. Эти другие атрибуты добавляются после разработки отношений.

Отметим, что диаграммы, приведенные на рисунках 7 и 8, иллюстрируют следующее условие применений сущностей *Продукт* и *Склад*: каждый продукт хранится не более чем на одном складе, и каждый склад хранит не более одного продукта.

В большинстве случаев для определения набора *отношений* проектируемой БД используются диаграммы *ER-типа*, а не диаграммы *ER-экземпляров*.

2.5. Характеристики связи

Важной характеристикой связи между двумя (и более) сущностями является *степень связи*.

Степень связи указывает на количество сущностей, охваченных данной связью. Связь *Хранится*, существующая между сущностями *Продукт* и *Склад*, называется *бинарной*, поскольку она связывает только две сущности. Связи более высокого порядка, существующие между *n* сущностями, называются *n-сторонними* (*n-арными*). Бинарные связи встречаются наиболее часто и во всех примерах настоящего пособия рассматриваются только бинарные связи.

Показатель кардинальности описывает количество возможных связей для каждого из экземпляров рассматриваемых сущностей.

Наиболее распространенными являются бинарные связи с показателями кардинальности «*один к одному*» (1 : 1), «*один ко многим*» (1 : *n*) или «*многие к одному*» (*n* : 1), «*многие ко многим*» (*m* : *n*).

Например, если для характеристики студентов вуза выделены две сущности *Анкетные данные студентов* и *Успеваемость студентов* (в каждой из этих сущностей ключевым атрибутом является шифр студента), то для связи *Учится* между экземплярами этих сущностей показатель кардинальности будет 1 : 1, т. е. каждый студент должен быть в точности один раз охарактеризован в каждой из этих сущностей.

Если рассмотреть две сущности *Академическая группа* (ключ *Шифр группы*) и *Факультет* (ключ *Шифр факультета*), то показатель кардинальности связи *Принадлежит* между экземплярами этих сущностей равен *n* : 1, т. е. каждая группа обязательно принадлежит какому-то одному факультету, но на каждом факультете есть несколько групп. Заметим, что между сущностями *Факультет* и *Академическая группа* показатель кардинальности связи равен 1 : *n*.

Для связи *Преподает* (*Читает*) между сущностями *Преподаватель* и *Учебный курс* (*Дисциплина*) показатель кардинальности будет *m* : *n*, так как каждый преподаватель читает несколько курсов и каждый курс может читаться разными преподавателями (для разных потоков).

Показатели кардинальности связей между экземплярами сущностей зависят прежде всего от производственных правил, установленных в данной организации, т. е. условиями применения сущностей.

Заметим, что показатель кардинальности не может полностью характеризовать связь между экземплярами сущности.

Пример 7. Пусть в одной организации действует следующее условие применения сущностей *Продукт* и *Склад*: каждый продукт хранится не более чем на одном складе, и каждый склад хранит не более одного продукта. Во второй организации аналогичное условие применения формулируется следующим образом: каждый продукт хранится не более чем на одном складе, и каждый склад хранит в точности один продукт. В обоих случаях связи имеют один показатель кардинальности 1 : 1, но действуют разные условия использования складов. Так, в первом случае каждый склад хранит в точности один продукт, а во втором случае на складе может и не быть продуктов, т. е. некоторый склад может не использоваться.

Чтобы различать связи такого рода, используется еще одна характеристика связи – *класс принадлежности сущности для* (конкретной) *связи*.

Если все экземпляры данной сущности должны участвовать в некоторой связи, то участие в этой связи называется *обязательным*, т. е. *класс принадлежности* данной сущности для этой связи является *обязательным*. Этот факт будет отмечен на диаграмме *ER-типа* жирной точкой на границе прямоугольника, обозначающего эту сущность.

Если экземпляры данной сущности могут не участвовать в некоторой связи, то участие в этой связи называется *необязательным*, т. е. *класс принадлежности* данной сущности для этой связи является *необязательным*. В этом случае на диаграмме *ER-типа* жирная точка вообще не помещается на диаграмме.

На рис. 9 приведена диаграмма *ER-типа* для связи *Хранится*, когда каждый продукт хранится не более чем на одном складе, и каждый склад хранит в точности один продукт.

Рис. 9. **Диаграмма *ER-типа* для сущностей с разными классами принадлежности**

Класс принадлежности сущности должен быть либо обязательным, либо необязательным. Он определяется правилами, регламентирующими деятельность предприятия. В некоторых случаях генерация отношений БД не зависит от класса принадлежности конкретной сущности для данной связи. В этих случаях говорят, что *класс принадлежности* данной сущности является *безразличным*.

Класс принадлежности сущности удобно определять по диаграмме *ER-экземпляров*: если может существовать точка (экземпляр сущности), из которой не выходит ни одна линия связи, то класс принадлежности данной сущности является необязательным.

2.6. Варианты связей

На примере связи *Хранится* для сущностей *Продукт* и *Склад* рассмотрим шесть вариантов характеристик бинарных связей. Однако следует заметить, что каждому из этих вариантов соответствует свое правило генерации отношений, которые будут рассмотрены далее.

Для сущностей *Продукт* и *Склад* могут действовать следующие альтернативные условия применения:

1. Каждый продукт хранится в точности на одном складе и каждый склад хранит в точности один продукт, соответствующие диаграммы *ER-экземпляров* и *ER-типа* приведены на рис. 10.

2. Каждый продукт хранится в точности на одном складе и каждый склад хранит не более одного продукта, соответствующие диаграммы *ER-экземпляров* и *ER-типа* приведены на рис. 11.

3. Каждый продукт хранится не более чем на одном складе и каждый склад хранит не более одного продукта. Соответствующие диаграммы *ER-экземпляров* и *ER-типа* приведены на рис. 12.

4. Каждый продукт может храниться на одном или большем числе складов либо вовсе не храниться на складах. Каждый склад хранит в точности один продукт. Соответствующие диаграммы *ER-экземпляров* и *ER-типа* приведены на рис. 13.

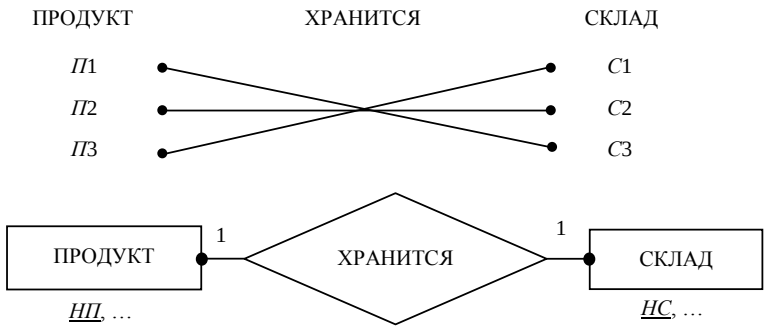


Рис. 10. **Диаграммы для бинарной связи с показателем кардинальности 1 : 1 и обязательным классом принадлежности каждой из сущностей**

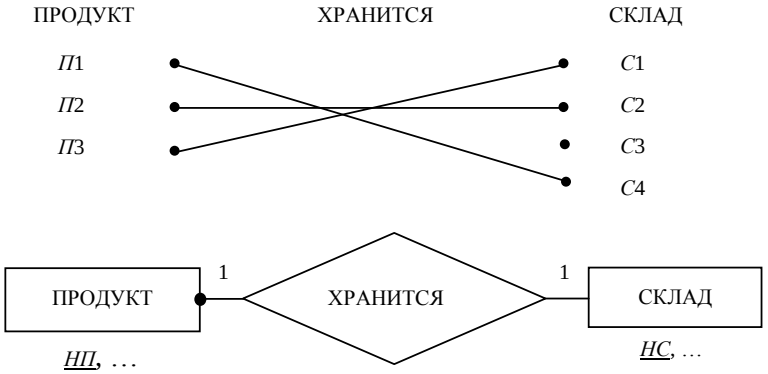


Рис. 11. **Диаграммы для бинарной связи с показателем кардинальности 1 : 1, класс принадлежности одной из сущностей, участвующих в связи (сущности *Продукт*), является обязательным, другой – необязательным**

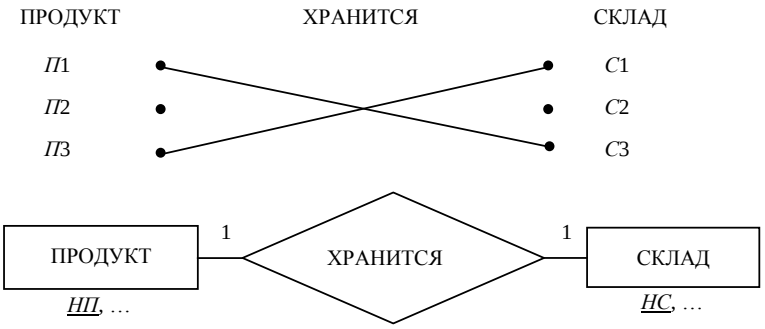


Рис. 12. **Диаграммы для бинарной связи с показателем кардинальности 1 : 1, класс принадлежности ни одной из сущностей не является обязательным**

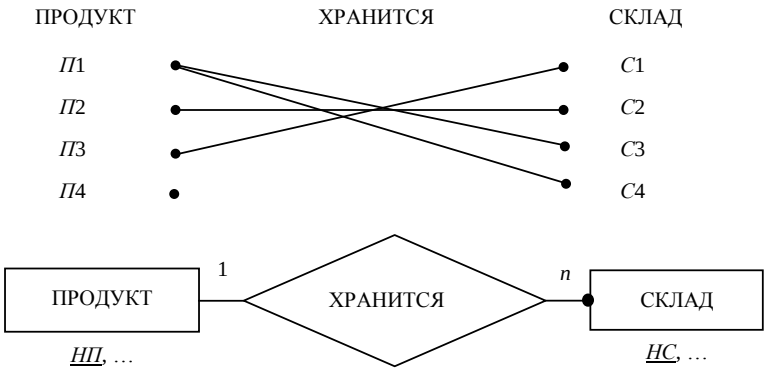


Рис. 13. **Диаграммы для бинарной связи с показателем кардинальности 1 : *n*, класс принадлежности сущности *Склад* является обязательным, а класс принадлежности сущности *Продукт* является безразличным**

5. Каждый продукт может храниться на одном или большем числе складов либо вовсе не храниться на складах, и каждый склад хранит не более одного продукта. Соответствующие диаграммы *ER-экземпляров* и *ER-типа* приведены на рис. 14.

6. Каждый продукт может храниться на одном или большем числе складов либо вовсе не храниться на складах и каждый склад хранит один или более продуктов либо вовсе не хранит продуктов. Соответствующие диаграммы *ER-экземпляров* и *ER-типа* приведены на рис. 15.

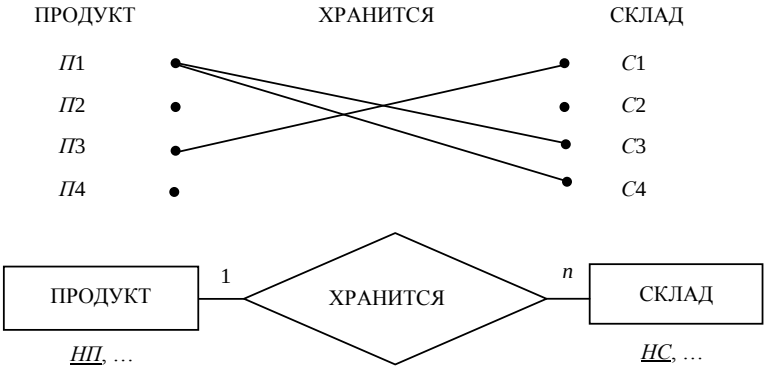


Рис. 14. **Диаграммы для бинарной связи с показателем кардинальности 1 : *n*, класс принадлежности *n*-связной сущности *Склад* не является обязательным, а класс принадлежности сущности *Продукт* безразличен**

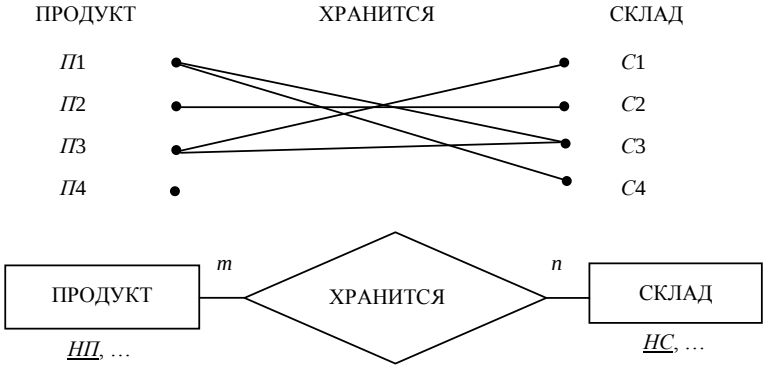


Рис. 15. **Диаграммы для бинарной связи с показателем кардинальности *m* : *n*, класс принадлежности сущностей безразличен**

2.7. Правила генерации отношений по диаграммам ER-типа

Как уже говорилось выше, рисунки 10–15 иллюстрируют все возможные виды бинарных связей, представляющих интерес с точки зрения необходимого количества формируемых отношений.

Этим перечисленным шести рисункам соответствуют шесть правил генерации отношений.

Правило 1. Если показатель кардинальности бинарной связи равен *1 : 1* и класс принадлежности обеих сущностей является обязательным, то требуется только *одно* отношение. Первичным ключом этого отношения может быть ключ любой из двух сущностей.

Таким образом, логическая модель БД для случая, изображенного на рис. 10, будет состоять из одного отношения. В данном отношении в качестве первичного ключа может быть использован любой из двух ключей *НП* или *НС*. Такое отношение можно описать следующим образом: *ХАР-ПРОД-СКЛ-РАСПР* (*НП, НАИМ, ЕИ, УП,…НС, ЕС, МОЛ,…*).

Многоточие в списке атрибутов показывает, что этот список, возможно, не полон, но отсутствующие атрибуты не влияют на данное рассмотрение отношения.

Примечание. Приведем еще два практических примера использования данного правила генерации отношений базы данных:

1. Таблица (сущность) *Товар* разбивается на две подтаблицы с одним и тем же ключом *Кодтов*, но с разными сведениями о товаре. Так, в первую подтаблицу заносятся основные сведения о товаре, во вторую – дополнительные.

2. Таблица (сущность) *Сотрудник* разбивается на две подтаблицы с одним и тем же ключом *ТабНом*, в одной из которых содержатся общедоступные сведения о сотрудниках предприятия, во второй – только конфиденциальные сведения о них.

В среде СУБД Access связи ссылочной целостности между такими подтаблицами будут иметь показатель кардинальности «*один к одному*».

Правило 2. Если показатель кардинальности бинарной связи равен *1 : 1* и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построение *двух* отношений. Под каждую сущность необходимо выделение одного отношения, при этом ключ сущности должен служить первичным ключом для соответствующего отношения. Кроме того, ключ сущности, для которой класс принадлежности является необязательным, добавляется в качестве атрибута в отношение, выделенное для сущности с обязательным классом принадлежности.

В соответствии с правилом 2 для случая, изображенного на рис. 11, логическая структура БД будет включать в себя два отношения, в первом из которых первичным ключом является атрибут *НП*, а во втором – *НС*:

- *ХАР-ПРОД-РАСПР* (*НП, НАИМ, ЕИ, УП, НС,…*);
- *ХАР-СКЛ* (*НС, ЕС, МОЛ,…*).

Первое отношение соответствует сущности *Продукт* и, кроме того, отражает данные о распределении продуктов по складам за счет включения в список атрибутов ключа *НС* сущности *Склад*. Таким образом, атрибут *НС* в первом отношении является внешним ключом.

Примечание. Приведем еще один часто встречающийся на практике пример использования данного правила генерации отношений базы данных. Имеется сущность *Персонал* с ключом *ТабНом*, в которой содержатся сведения общего характера: адрес, телефон, дата рождения и т. д. Среди работников могут выделяться отдельные «ролевые» группы: мастера, рабочие и др. В этом случае вводятся дополнительные сущности *Мастер* и *Рабочий* с ключами *КодМаст* и *КодРаб*, в которых содержится специфическая информация. Например, для мастера – номер служебного телефона, надбавка за ненормированный рабочий день и т. д., а для рабочего – разряд, почасовая ставка и т. д. Тогда с использованием данного правила отношения будут сгенерированы следующим образом:

- Персонал (*ТабНом*, …);
- Мастер (*КодМаст*, …, *ТабНом*);
- Рабочий (*КодРаб*, …, *ТабНом*).

Ключи сгенерированных отношений подчеркнуты.

В среде СУБД Access связи ссылочной целостности по полю *ТабНом* между таблицей *Персонал* и таблицей *Мастер*, а также между таблицей *Персонал* и таблицей *Рабочий* будут «один к одному».

Правило 3. Если показатель кардинальности бинарной связи равен *1 : 1* и класс принадлежности ни одной сущности не является обязательным, то необходимо использовать *три* отношения: по одному для каждой сущности, ключи которых служат в качестве первичных в соответствующих отношениях, и одного для связи. Среди своих атрибутов отношение выделяемой связи будет иметь по одному ключу каждой сущности.

В соответствии с правилом 3 для случая, изображенного на рис. 12, логическая структура БД будет включать в себя три отношения:

- *ХАР-ПРОД* (*НП, НАИМ, ЕИ, УП, …*);
- *ХАР-СКЛ* (*НС, ЕС, МОЛ, …*);
- *РАСПР* (*НП, НС, …*).

Первое из этих отношений соответствует сущности *Продукт* и содержит характеристики продуктов, первичным ключом является атрибут *НП*. Второе отношение соответствует сущности *Склад* и содержит характеристики имеющихся складов, первичный ключ – *НС*.

Третье отношение соответствует связи и характеризует распределение продуктов по складам. Здесь среди атрибутов должны быть ключи сущностей *Продукт* и *Склад*, *НП* и *НС*, соответственно.

Примечание. Приведем еще один пример использования данного правила генерации отношений базы данных. Имеются сущности *Мужчина* с ключом *КодМ* и *Женщина* с ключом *КодЖ*. Некоторые пары вступают в брак. Предполагается, что браки моногамные и не все мужчины и женщины состоят в браке. Тогда с использованием данного правила отношения будут сгенерированы следующим образом:

• Мужчина (КодМ, …);
• Женщина (КодЖ, …);
• СвидетельствоОБраке (НомерСвидетельства, Дата, МестоРегистрации,…, КодМ, КодЖ).

Ключи сгенерированных отношений подчеркнуты.

В среде СУБД Access связи ссылочной целостности между таблицей *СвидетельствоОБраке* и таблицей *Мужчина* по полю *КодМ*, а также между таблицей *СвидетельствоОБраке* и таблицей *Женщина* по полю *КодЖ* будут «*один к одному*».

Правило 4. Если показатель кардинальности бинарной связи равен *1 : n* и класс принадлежности *n*-связной сущности является обязательным, то достаточным является использование *двух* отношений, по одному на каждую сущность, при условии, что ключ сущности каждой сущности служит в качестве первичного ключа для соответствующего отношения. Дополнительно ключ односвязной сущности должен быть добавлен как атрибут в отношение, отводимое *n*-связной сущности.

В соответствии с правилом 4 для случая, изображенного на рис. 13, логическая структура БД будет включать в себя два отношения, в первом из которых первичным ключом является атрибут *НП*, а во втором – *НС*, при этом во второе отношение надо добавить в качестве внешнего ключа атрибут *НП*:

• *ХАР-ПРОД* (*НП, НАИМ, ЕИ, УП*,…);
• *ХАР-СКЛ-РАСПР* (*НС, ЕС, МОЛ*,…, *НП*).

Правило 5. Если показатель кардинальности бинарной связи равен *1 : n* и класс принадлежности *n*-связной сущности является необязательным, то необходимо формирование *трех* отношений: по одному для каждой сущности, причем ключ каждой сущности служит первичным ключом соответствующего отношения, и одного отношения для связи. Связь должна иметь среди своих атрибутов ключ сущности от каждой сущности.

В соответствии с правилом 5 для случая, изображенного на рис. 14, логическая структура БД будет включать в себя три отношения, в первом из которых первичным ключом является атрибут НП, во втором – *НС*. В отношении для связи среди атрибутов должны быть ключи сущностей *Продукт* и *Склад*, *НП* и *НС* соответственно. Таким образом, в рассматриваемом случае логическая структура БД будет аналогична модели, сформированной в соответствии с правилом 3:

• *ХАР-ПРОД* (*НП, НАИМ, ЕИ, УП*,…);
• *ХАР-СКЛ* (*НС, ЕС, МОЛ*,…);
• *РАСПР* (*НП, НС*,…).

Правило 6. Если показатель кардинальности бинарной связи равен *m : n*, то для хранения данных необходимо *три* отношения: по одному для каждой сущности, причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения, и одного отношения для связи. Последнее отношение должно иметь в числе своих атрибутов ключ каждой сущности.

В соответствии с правилом 6 логическая структура БД для случая, изображенного на рис. 15, будет полностью аналогична предыдущему случаю:

• *ХАР-ПРОД* (*НП, НАИМ, ЕИ, УП*,…);
• *ХАР-СКЛ* (*НС, ЕС, МОЛ*,…);
• *РАСПР* (*НП, НС*,…).

На практике при проектировании баз данных для экономико-социальной области применения наиболее часто используется правило 4, т. е. сущности, отображаемые в БД, ассоциированы между собой связью с показателем кардинальности 1 : *n*, причем *n*-связная сущность имеет обязательный класс принадлежности.

2.8. Особенности ER-метода для экономических приложений

Особенностью использования *ER-метода* логического проектирования баз данных для экономических приложений является то, что часто в качестве сущностей рассматриваются экономические документы, которые имеют заголовочную и табличную части.

Если в результате концептуального проектирования БД были выделены документы-сущности, то перед созданием диаграммы *ER-типа* целесообразно выполнить дополнительные действия, облегчающие осмысленное применение упомянутых шести правил генерации отношений проектируемой БД.

Сущность, представляющую собой документ с заголовочной и табличной частями, необходимо представить в виде двух сущностей – *Заголовок документа* и *Строка документа*. Между этими сущностями устанавливается бинарная связь, которую можно назвать *Объединяются*. Показатель кардинальности этой связи равен 1 : *n* и *n*-связная сущность *Строка документа* имеет обязательный класс принадлежности. Поэтому для этой пары сущностей вместе с их связью должны генерироваться два отношения (по правилу 4) – по одному на каждую сущность. При этом ключевой атрибут каждой сущности будет первичным ключом для соответствующего отношения. Дополнительно ключ односвязной сущности *Заголовок документа* должен быть добавлен как атрибут в отношение, отводимое *n*-связной сущности *Строка документа*.

Все вышесказанное можно назвать правилом генерации отношений для сущностей, представляющих собой экономический документ с заголовочной и табличной частями.

В качестве примера рассмотрим документ, представленный в табл. 15.

Таблица 15. **Акт № 175 от 12.02.2010 г. передачи материальных ценностей** Грушину А. Б. (ФИО МОЛ)

Плановый отдел (подразделение)

Инвентарный номер	Название инвентарного объекта	Наименование инвентарной группы	Балансовая стоимость, р.
ИПО123	Принтер LBP-810	Вычислительная техника	520000
ИПО348	Стул	Мебель	18050
ИПО349	Кресло	Мебель	36800

Сущность *Акт* имеет в заголовочной части следующие атрибуты: *Номер акта*, *Дата акта*, *ФИО МОЛ*, *Подразделение*, а в табличной части – *Инвентарный номер*, *Название инвентарного объекта*, *Наименование инвентарной группы*, *Балансовая стоимость*. Заметим, что атрибут *Номер акта* является ключом сущности.

На основании сформулированного выше правила генерации отношений для экономических документов с целью упрощения процесса проектирования БД целесообразно представить сущность *Акт* в виде двух сущностей – *Заголовок акта* и *Строка акта*, которые выглядят следующим образом:

- *Заголовок акта* (*Номер акта*, *Дата акта*, *ФИО МОЛ*, *Подразделение*);
- *Строка акта* (*Инвентарный номер*, *Название инвентарного объекта*, *Наименование инвентарной группы*, *Балансовая стоимость*, *Номер акта*).

Как и для всякого экономического документа, можно считать, что между сущностями *Акт* и *Строка акта* установлена связь *Объединяются*. Эта связь имеет показатель кардинальности $1:n$, классы принадлежности обеих сущностей являются обязательными. Соответствующие диаграммы *ER-экземпляров* и *ER-типа* для связи *Объединяются* приведены на рис. 16.

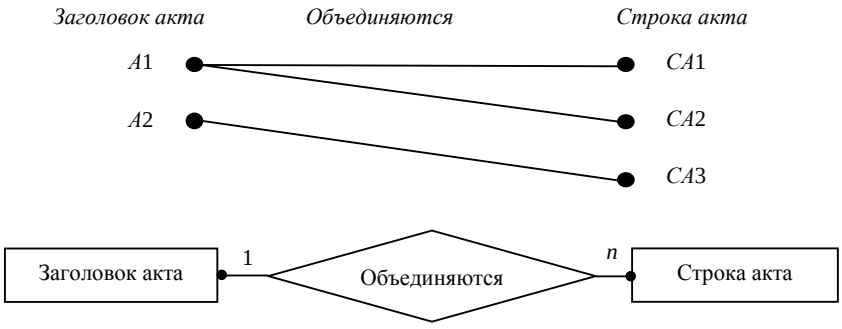


Рис. 16. **Диаграммы для связи *Объединяются***

Преимуществом представления документа в качестве двух сущностей является то, что для большинства связей будет применимо четвертое правило генерации отношений.

2.9. Методика применения ER-метода

В качестве исходных данных для выполнения логического проектирования БД на этапе концептуального проектирования должно быть определено следующее:

- набор сущностей;
- предварительный перечень атрибутов для каждой сущности (основные атрибуты);
- ключевой атрибут для каждой сущности;
- набор связей между сущностями;
- описание условий применения объектов (сущностей) на данном предприятии.

ER-метод логического проектирования предполагает выполнение следующего:

- если в результате концептуального проектирования БД были выделены документы-сущности, то необходимо каждую из них представить в виде двух сущностей: *Заголовок документа* и *Строка документа*;
- определение показателя кардинальности для каждой из связей (на основе производственных условий использования сущностей и, если это необходимо, диаграммы *ER-экземпляров*);
- определение класса принадлежности каждой сущности в каждой связи;
- построение диаграммы *ER-типа*;
- определение всех отношений БД, их атрибутов и первичных ключей на основании правил 1–6.

Также следует отметить роль диаграмм *ER-типа* и *ER-экземпляров* при проектировании базы данных.

Построение диаграммы *ER-типа* при проектировании БД дает возможность выбрать правило генерации отношений.

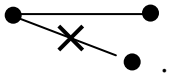
Кроме того, совокупность диаграмм *ER-типа* для всех связей проектируемой БД представляет собой компактное и наглядное представление структуры БД. Такое изображение структуры БД полезно как для предварительного знакомства с БД, так и для процесса ее модификации.

Диаграмма *ER-экземпляров* дает возможность определить показатель кардинальности связи и класс принадлежности каждой сущности для рассматриваемой связи. Для того, чтобы построить диаграмму необходимо выполнить следующее:

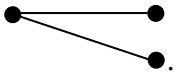
- Определить, сколько линий (одна или больше) может исходить из точки в левой части диаграммы.
- Определить, сколько линий (одна или больше) может входить в точку правой части диаграммы.
- Уточнить, может ли в левой части диаграммы быть точка, из которой не выходит ни одна линия.
- Определить, может ли в правой части диаграммы быть точка, в которую не входит ни одна линия.

При этом действуют следующие правила:

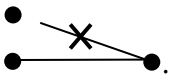
1. Если точка в левой части диаграммы может быть соединена только с одной точкой в правой части, то связь будет иметь показатель … : 1:



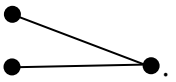
2. Если точка в левой части диаграммы может быть соединена с несколькими точками в правой части, то связь будет иметь показатель … : *n*:



3. Если точка в правой части диаграммы может быть соединена только с одной точкой в левой части, то связь будет иметь показатель 1 : …:

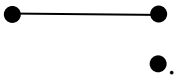


4. Если точка в правой части диаграммы может быть соединена с несколькими точками в левой части, то связь будет иметь показатель *n* : …:



5. Если в левой части диаграммы может существовать точка, несвязанная ни с одной точкой в правой части, то класс принадлежности левой сущности является необязательным:

6. Если в правой части диаграммы может существовать точка, несвязанная ни с одной точкой в левой части, то класс принадлежности правой сущности является необязательным:



3. СОЗДАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ И РАБОТА С НЕЙ В СРЕДЕ СУБД ACCESS (ЛАБОРАТОРНЫЙ ПРАКТИКУМ)

3.1. Общие положения

Целью данного лабораторного практикума является выработка практических навыков по созданию реляционной базы данных и работе с ней.

Для достижения этой цели на примере простой БД *Учет заказов* рассмотрены все этапы работы с базами данных – от проектирования БД до ее использования (эксплуатации). Последовательно решаются следующие задачи:

1. Концептуальное и логическое проектирование БД.
1.1. Уточнение набора сущностей и связей между ними.
1.2. Построение диаграммы *ER-типа* для проектируемой БД.
1.3. Определение состава таблиц БД и состава полей каждой таблицы.
2. Физическое проектирование БД в среде СУБД Access.
2.1. Создание и определение структур таблиц базы данных.
2.2. Установление связей между таблицами (создание схемы данных).
3. Уточнение структуры таблиц: автоматическая подстановка значений внешних ключей.

3.1. Заполнение БД.
3.2. Создание форм.
4. Ввод данных в таблицы.
4.1. Использование БД.
4.2. Создание запросов.
Формирование отчета о продажах за анализируемый период.

Задача 1 решается без компьютера. Задачи 2–4 решаются на компьютере в среде СУБД Access и рассчитаны на 4 часа аудиторных занятий.

3.2. Концептуальное и логическое проектирование

3.2.1. Результат концептуального проектирования

Предполагается, что создаваемая реляционная БД *Учет заказов* используется в фирме, торгующей продовольственными товарами. Эта БД предназначена для хранения сведений о клиентах и товарах, а также информации обо всех выполненных заказах клиентов на покупку товаров.

Будем предполагать в нашем случае, что на этапе концептуального проектирования выделены три сущности *Клиент*, *Товар* и *Заказ*.

Сущность *Клиент* характеризуется следующими атрибутами: код клиента (*КодКлиента*), наименование клиента (*НаимКлиент*), контактное лицо (*КонтактЛицо*), адрес клиента (*Адрес*). Ключевой атрибут сущности *Клиент* – атрибут *КодКлиента*.

Сущность *Товар* характеризуется такими атрибутами, как код товара (*КодТов*), наименование товара (*НаимТовар*), единица измерения (*ЕдИзм*), цена (*Цена*), исходное количество товара в фирме на начало отчетного периода (*ИсхКолич*). Ключевой атрибут сущности *Товар* – атрибут *КодТов*.

Сущность *Заказ* содержит сведения о бумажном документе – бланке заказа (рис. 17). Этот документ имеет заголовочную («шапку») и табличную части. В шапке документа содержатся следующие атрибуты: номер заказа (*НомерЗаказа*), дата заказа (*ДатаЗаказа*), наименование клиента (*НаимКлиент*). Табличная часть документа представляет собой набор строк, в каждой из которых содержатся такие атрибуты, как наименование товара (*НаимТовар*) и количество (*Количество*). Ключевой атрибут сущности *Заказ* – атрибут *НомерЗаказа*.

НомерЗаказа **5** *ДатаЗаказа* **04.12.2009**
Наименование клиента **Пьер и К**

Наименование товара	*Количество*
Желе апельсиновое	**4**
Желе виноградное	**34**
Желе вишневое	**13**

Рис. 17. **Пример бланка заказа**

Сущности *Клиент* и *Заказ* ассоциированы связью *Оформлен*, а сущности *Заказ* и *Товар* связью *Включен*.

Необходимо учесть следующие обстоятельства (условия применения):
- номера заказов не повторяются на протяжении всего периода учета;
- в один день может быть оформлено несколько заказов, причем с одним и тем же клиентом;
- в одну строку табличной части заказа может быть включен только один товар;
- один товар не включается в две строки одного бланка заказа;
- один и тот же товар может быть включен в табличные части разных заказов.

3.2.2. Уточнение набора сущностей и связей между ними

Документу *Заказ*, как и всякому экономическому документу с заголовочной и табличной частями, удобно поставить в соответствие две сущности *Заказ* (заголовок заказа) и *Строка заказа*.

Сущность *Заказ* имеет атрибуты, соответствующие реквизитам заголовочной части документа: *НомерЗаказа*, *ДатаЗаказа*, *НаимКлиент*.

Сущность *Строка заказа* имеет атрибуты *НаимТовар* и *Количество*.

Как и для всякого экономического документа, можно считать, что между сущностями *Заказ* и *Строка заказа* установлена связь *Объединяются*.

По условию задачи сущности *Клиент* и *Заказ* ассоциированы связью *Оформлен*. Так как нами разделена сущность *Заказ* на две сущности *Заказ* и *Строка заказа*, то необходимо уточнить, с какой из этих двух сущностей связана сущность *Клиент*. Ясно, что следует рассматривать связь *Оформлен* между сущностями *Клиент* и *Заказ*, так как клиент упоминается именно в заголовочной части документа.

Аналогично уточним, что связь *Включен* ассоциирована между сущностями *Товар* и *Строка заказа*, так как товар упоминается именно в табличной части документа

3.2.3. Построение диаграммы ER-типа для проектируемой базы данных

Построим сначала для каждой связи диаграмму *ER-экземпляров*.

При построении диаграммы *ER-экземпляров* для связи *Объединяются* надо исходить из следующего:

- одна заголовочная часть заказа может объединять несколько строк документа;
- одна конкретная строка заказа может находиться только в одном конкретном документе;
- не может существовать заказ, в котором есть заголовочная часть и нет ни одной строки в табличной части документа;
- не может существовать заказ, в котором есть строка табличной части и отсутствует заголовочная часть.

Следовательно, рассматриваемая связь имеет показатель кардинальности *1 : n*, а классы принадлежности обеих сущностей являются обязательными. Таким образом, получаем диаграмму *ER-экземпляров*, приведенную на рис. 18.

При построении диаграммы *ER-экземпляров* для связи *Оформлен* надо исходить из того, что:

- один клиент может оформить несколько заказов;
- один конкретный заказ может оформить только один клиент;
- не может существовать заказа, который не оформлен каким-либо клиентом;
- может существовать клиент, который в рассматриваемый период не оформил ни один заказ.

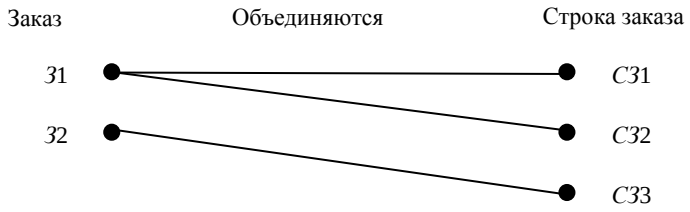


Рис. 18. **Диаграмма *ER-экземпляров* для связи *Объединяются***

Таким образом, связь *Оформлен* имеет показатель кардинальности *1 : n*, класс принадлежности сущности *Заказ* является обязательным, а сущности *Клиент* – необязательным. Таким образом, получаем диаграмму *ER-экземпляров*, приведенную на рис. 19.

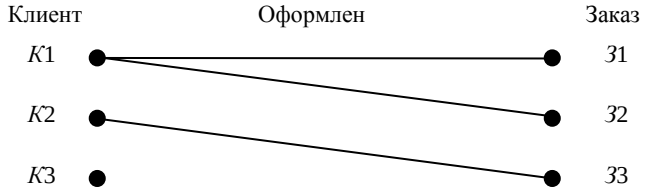


Рис. 19. **Диаграмма *ER-экземпляров* для связи *Оформлен***

При построении диаграммы *ER-экземпляров* для связи *Включен* надо исходить из следующего:

- в одну строку заказа может быть включен только один товар;
- один и тот же товар может быть включен в несколько строк различных заказов;
- не может существовать строки заказа, в которую не включен товар;
- может существовать товар, который не включен ни в одну строку ни одного заказа.

Таким образом, связь *Включен* имеет показатель кардинальности *n : 1*, класс принадлежности сущности *Строка заказа* является обязательным, а класс принадлежности сущности *Товар* – необязательным. Соответствующая диаграмма приведена на рис. 20.

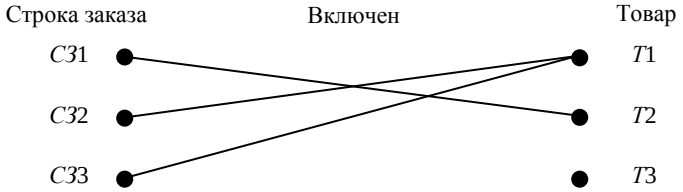


Рис. 20. **Диаграмма *ER-экземпляров* для связи *Включен***

Обратите внимание, что все связи имеют один и тот же показатель кардинальности, равный *1 : n*. Это наиболее типичный случай при проектировании баз данных. После того, как с помощью диаграмм *ER-экземпляров* определен для каждой связи показатель кардинальности и класс принадлежности сущностей, ассоциированных связью, можно построить диаграмму *ER-типа* для проектируемой базы данных *Учет заказов*. Искомая диаграмма приведена на рис. 21.

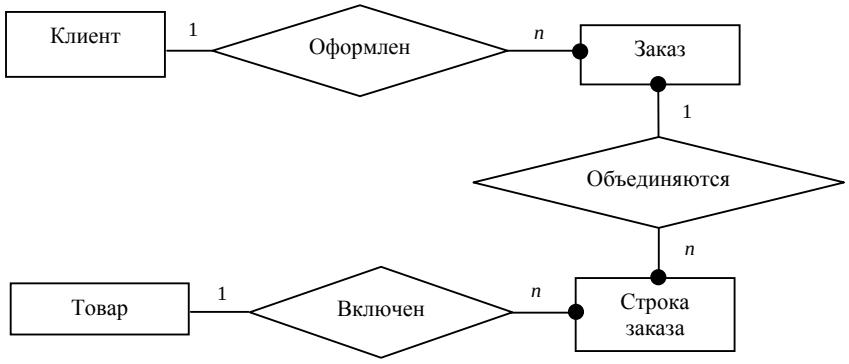


Рис. 21. **Диаграмма *ER-типа* проектируемой базы данных**

3.2.4. Определение состава таблиц базы данных и состава полей каждой таблицы

На основании четвертого правила генерации отношений связь *Оформлен* порождает два отношения по одному для каждой сущности, причем ключевой атрибут *КодКлиента* сущности *Клиент* должен быть включен в число атрибутов отношения *Заказ*. После включения атрибута *КодКлиента* наличие атрибута *НаимКлиент* в отношении *Заказ* становится избыточным, так как значение указанного атрибута однозначно определяется значением атрибута *КодКлиента*. Получаем следующие отношения:

- *Клиент* (*КодКлиента, НаимКлиент, КонтактЛицо, Адрес*);
- *Заказ* (*НомерЗаказа, ДатаЗаказа, КодКлиента*).

На основании четвертого правила генерации отношений связь *Объединяются* порождает два отношения по одному для каждой сущности, причем ключевой атрибут *НомерЗаказа* сущности *Заказ* должен быть включен в число атрибутов отношения *СтрокаЗаказа.* Таким образом, к сгенерированным отношениям добавляется отношение: *СтрокаЗаказа* (*НомерЗаказа, НаимТовар, Количество*).

На основании четвертого правила генерации отношений связь *Включен* порождает два отношения по одному для каждой сущности, причем ключевой атрибут *КодТов* сущности *Товар* должен быть включен в число атрибутов отношения *СтрокаЗаказа.* После включения атрибута *КодТов* наличие атрибута *НаимТовар* в отношении *СтрокаЗаказа* становится избыточным, так как значение указанного атрибута однозначно определяется значением атрибута *КодТов*. Следовательно, получаем следующие отношения:

- *Товар* (*КодТов, НаимТовар, ЕдИзм*, *Цена*, *ИсхКолич*);
- *СтрокаЗаказа* (*НомерЗаказа*, *КодТов, Количество*).

Таким образом, искомая БД состоит из четырех отношений:

- *Клиент* (***КодКлиента****, НаимКлиент, КонтактЛицо, Адрес*);
- *Товар* (***КодТов***, *НаимТовар, ЕдИзм*, *Цена*, *ИсхКолич*);
- *Заказ* (***НомерЗаказа****, ДатаЗаказа, КодКлиента*);
- *СтрокаЗаказа* (***НомерЗаказа***, ***КодТов****, Количество*).

У таблиц *Клиент*, *Товар* и *Заказ* первичные ключи (выделены полужирным шрифтом) простые и состоят из одного поля.

У таблицы *СтрокаЗаказа* первичный ключ – составной и состоит из двух полей *НомерЗаказа* и *КодТов*. Эти два поля вместе однозначно определяют бланк заказа (поле *НомерЗаказа*) и строку в его табличной части (поле *КодТов*). Следует помнить, что один товар не указывается в двух строках одного бланка заказа.

3.3. Физическое проектирование базы данных

3.3.1. Запуск СУБД Access

Физическое проектирование базы данных производится в среде выбранной реляционной СУБД – в среде СУБД Access.

Примечание. Дальнейшее изложение ведется для версии Access-2000, но отличия от других версий (Access-95, Access-97) – незначительны.

В последующем изложении перемежаются фрагменты текста, поясняющие смысл выполняемых действий, с фрагментами, представляющими собой инструкции о порядке выполнения действий на компьютере.

Фрагменты-инструкции предваряются пометкой *Задание №* и завершаются пометкой *Конец задания*. Задания нумеруются последовательно в пределах заданий 2–4 практикума.

Задание 1. Создайте файл базы данных в папке своей группы. Для этого выполните следующее:

1. В папке *D:\Stud* создайте папку своей группы, например, *Бс21з*.
2. Загрузите СУБД Access (проще всего щелчком по соответствующей кнопке на панели MS Office).
3. Переключатель *Создание базы данных* установите в значение *Новая база данных*, нажмите на кнопку *ОК*. Появится диалоговое окно *Файл новой базы данных*.
4. Выберите папку с именем группы (например, *D:\Stud\Бс21з*) для сохранения базы данных.
5. Присвойте файлу базы данных имя *УчетЗаказов* с добавленной фамилией – *УчетЗаказовИванов*. Нажмите кнопку *Создать*.

Примечание. Если за компьютером работают два студента, то допустимо такое название файла базы данных: *УчетЗаказовИвановПетров.mdb*. Расширение *mdb* к имени файла БД приформировывается СУБД Access автоматически.

После выполнения задания 1 появится окно *УчетЗаказов: база данных* (или, например, *УчетЗаказовИванов: база данных*), которое не закрывается в течение всего сеанса работы с данной базой. Оно содержит все объекты базы данных: *Таблицы*, *Запросы*, *Формы*, *Отчеты*, *Макросы*, *Модули*.

Конец задания.

3.3.2. Создание и определение структур таблиц базы данных

Определение структуры таблицы заключается в том, что для каждого его поля задаются:

- имя поля;
- тип данных, которые будут содержаться в данном поле;
- описание (комментарий о назначении поля);
- общие свойства поля (размер, формат, число десятичных знаков, маска ввода, подпись, значение по умолчанию, условие на значение, сообщение об ошибке).

При создании таблиц следует придерживаться следующей последовательности. Вначале создаются таблицы с нормативно-справочной информацией (*Товар, Клиент*), затем – с оперативно-учетной информацией (*Заказ, СтрокаЗаказа*). Если таблицы с оперативно-учетной информацией соответствуют документу с шапкой и табличной частью, то вначале создается таблица для шапки документа (*Заказ*), затем – для табличной части (*СтрокаЗаказа*).

Задание 2. Создайте таблицу *Товар*. Для этого выполните следующее:

1. Сделайте активной вкладку *Таблицы* и нажмите на кнопку *Создать*. В диалоговом окне *Новая таблица* выберите способ создания – *Конструктор,* нажмите на кнопку *ОК*. В результате откроется окно *Конструктора* таблиц, состоящее из двух частей.

В верхней части окна для каждого поля таблицы вводится *Имя поля* с клавиатуры; выбирается *Тип данных* из раскрывающегося списка, вводится *Описание* поля (не обязательно).

В нижней части окна можно просмотреть и изменить свойства текущего поля. Текущее поле выделяется маркером текущего поля ▸ или ▶.

2. Полям таблицы *Товар* придайте свойства в соответствии с рис. 22.

Примечание. Обратите внимание, имена полей написаны без пробелов и каждое слово пишется с большой буквы. Кроме того, для простоты в данной таблице *Товар* и во всех остальных таблицах базы данных все текстовые поля кроме кодов имеют размер 50 (символов), а все числовые поля имеют размер длинного целого. Поля, содержащие коды, как правило, объявляют текстовыми небольшой длины.

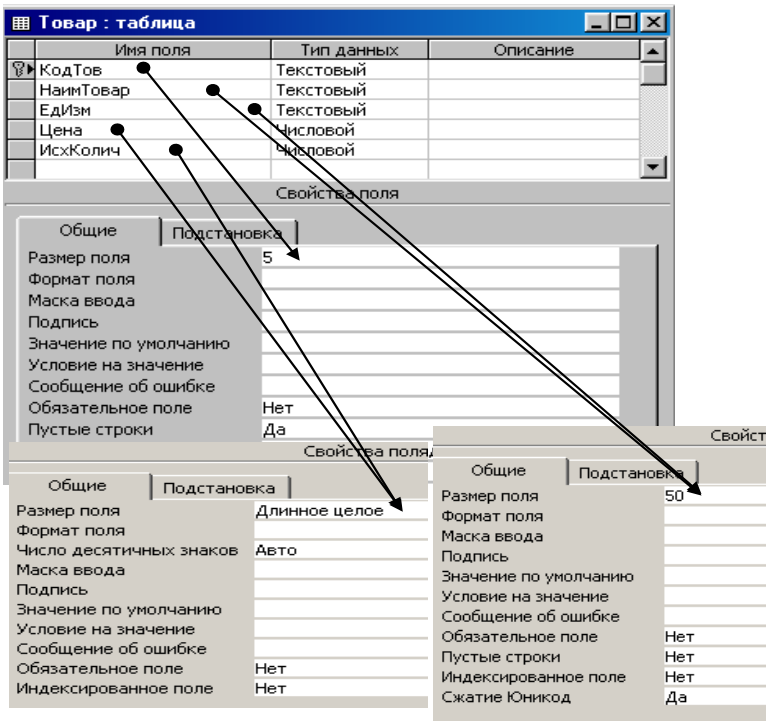


Рис. 22. ***Конструктор таблиц*****. Поля таблицы *Товар* и их свойства**

3. Полю *КодТов* придайте свойство первичного ключа таблицы *Товар*. Для этого *выделите* поле *КодТов* и щелкните на панели инструментов по кнопке с изображением ключа. Слева от поля *КодТов* должен появиться маркер ключевого поля  .

Примечание. *Выделение поля* осуществляется щелчком мыши по *области выделения строки* нужного поля. *Область выделения строки* – это прямоугольник с левого края строки, при щелчке которого выделяется вся строка.

4. Сохраните таблицу. Для этого выполните щелчок по кнопке *Сохранить* на панели инструментов. В открывшемся диалоговом окне *Сохранение* введите имя таблицы *Товар*, нажмите на кнопку *ОК*. Закройте окно конструктора таблиц (его имя сейчас *Товар: таблица*).

Конец задания.

Задание 3. Аналогично создайте таблицу *Клиент*. Все поля таблицы являются текстовыми.

Первичный ключ – *КодКлиента* (его свойства совпадают со свойствами поля *КодТов* таблицы *Товар*).

Свойства полей *НаимКлиент*, *КонтактЛицо* и *Адрес* совпадают со свойствами поля *НаимТовар* таблицы *Товар*.

Описание полей можно не вводить.

Конец задания.

Задание 4. Создайте таблицу *Заказ*.

Первичный ключ – *НомерЗаказа* (его свойства совпадают со свойствами поля *КодТов* таблицы *Товар*).

Примечание. Для убыстрения ввода значений поля *НомерЗаказа* в свойство *Значение по умолчанию* можно (нужно) добавить префикс *Зак* номера заказа.

Поле *КодКлиента* – текстовое (его свойства совпадают со свойствами поля *КодКлиента* таблицы *Клиент*, кроме свойства ключевого поля).

Поле *ДатаЗаказа* имеет тип данных *Дата/время*, формат – *Краткий формат даты*.

Конец задания.

Задание 5. Создайте таблицу *СтрокаЗаказа*.

Первичный ключ – составной и состоит из полей *НомерЗаказа* и *КодТов* (их свойства совпадают со свойствами полей *НомерЗаказа* таблицы *Заказ* и *КодТов* таблицы *Товар* соответственно).

Примечание. Если ключ составной, то перед щелчком по кнопке с изображением ключа необходимо *выделить все поля*, которые требуется определить как ключевые.

Выделение нескольких полей осуществляется щелчками при нажатой клавише *Ctrl* по области выделения строки для каждого поля.

Поле *Количество* имеет числовой тип данных и размер длинного целого (его свойства совпадают со свойствами поля *Цена* таблицы *Товар*).

Конец задания.

3.3.3. Установление связей между таблицами (создание схемы данных)

У всех таблиц в нашей базе данных имеется хотя бы одно поле, которое есть в какой-либо другой таблице БД (общие поля), так у таблиц *Клиент* и *Заказ* общее поле – *КодКлиента*; у таблиц *Заказ* и *СтрокаЗаказа* – *НомерЗаказа*; у таблиц *Товар* и *СтрокаЗаказа* – *КодТов*.

Именно по этим общим полям устанавливаются связи между таблицами БД.

В нашей БД каждое общее поле является первичным ключом только в одной таблице. Следовательно, в другой таблице оно является *внешним* ключом. Таким общим полям соответствуют связи типа «один ко многим», т. е. одна из связываемых таблиц является главной в этой связи, другая – подчиненной. Главной является таблица, для которой общее поле является первичным ключом. Таким образом, в первой связи главной является таблица *Клиент*, во второй – *Заказы*, в третьей – таблица *Товар*.

Задание 6. Создайте схему данных БД. Для этого выполните следующее:

1. Нажмите на кнопку *Схема данных* на панели инструментов. Появится окно *Схема данных* и в нем окно *Добавление таблицы*. Окно *Добавление таблицы* переместите вниз («схватив» за заголовок – *drag&drop*) для увеличения обзора.

2. Выделите таблицу *Товар* и нажмите на кнопку *Добавить*. Аналогично добавьте остальные три таблицы: *СтрокаЗаказа*, *Заказ* и *Клиент*. Закройте окно *Добавление таблицы*.

Примечание. В окне *Схема данных* четыре таблицы. Между ними устанавливаются три связи:
- связь между таблицами *Клиент* и *Заказ* по полю *КодКлиента*;
- связь между таблицами *Заказ* и *СтрокаЗаказа* по полю *НомерЗаказа*;
- связь между таблицами *Товар* и *СтрокаЗаказа* по полю *КодТов*.

В первой связи главной является таблица *Клиент*, во второй – *Заказ*, а в третьей – таблица *Товар*.

3. Процедура установления отдельной связи следующая:
- В окне *Схема данных* перетащите (*drag&drop*) поле связи из главной таблицы на аналогичное поле в подчиненной таблице.
- Откроется окно диалога *Связи*, в котором установите флажки *Обеспечение целостности данных* и *Каскадное обновление связанных полей*. Флажок *Каскадное удаление связанных полей* устанавливать не рекомендуется.
- Нажмите на кнопку *Создать*.

4. Процедуру, указанную в пункте 3, выполните для всех трех связей.

5. Закройте и сохраните *Схему данных*.

На рис. 23 приведена схема данных БД *Учет заказов*.

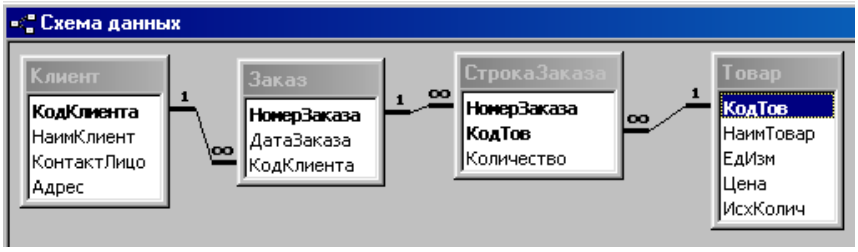


Рис. 23. **Схема данных БД *Учет заказов***

Конец задания.

3.3.4. Уточнение структуры таблиц: автоматическая подстановка значений внешних ключей

В связях типа «*один ко многим*» (все связи в схеме данных БД *Учет заказов* имеют данный тип) удобно значения внешнего ключа в подчиненной таблице получать из главной таблицы путем автоматической подстановки значения первичного ключа выбранной записи главной таблицы.

Задание 7. Для автоматической подстановки значений внешнего ключа *КодКлиента* в таблицу *Заказ* из таблицы *Клиент* выполните следующее:

• Откройте таблицу *Заказ* в режиме *Конструктор*. Выделите поле *КодКлиента*.
• Выполните щелчок по вкладке *Подстановка* в *Свойствах поля*.
• Установите свойства подстановки в соответствии с рис. 24. Для ускорения ввода используйте кнопки раскрывающихся списков, появляющиеся при выборе некоторых строк свойств.

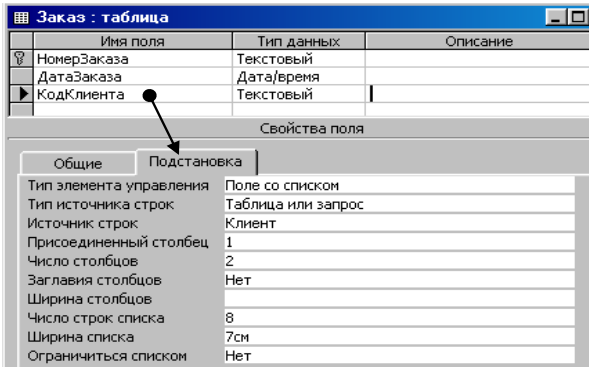


Рис. 24. **Свойства подстановки поля *КодКлиента* из таблицы *Клиент* в таблицу *Заказ***

Конец задания.

Задание 8. Для автоматической подстановки значений внешнего ключа *КодТов* в таблицу *СтрокаЗаказа* из таблицы *Товар* выполните следующее:
• Откройте таблицу *СтрокаЗаказа* в режиме *Конструктор*. Выделите поле *КодТов*.
• Выполните щелчок по вкладке *Подстановка* в *Свойствах поля*.
• Установите свойства подстановки в соответствии с рис. 25.

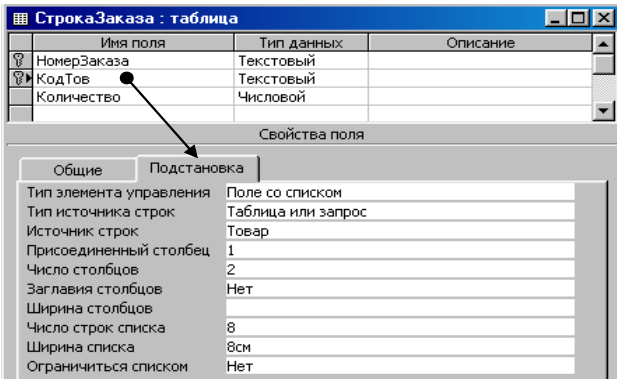


Рис. 25. **Свойства подстановки поля *КодТов* из таблицы *Товар* в таблицу *СтрокаЗаказа***

Конец задания.

Задание 9. Для автоматической подстановки значений внешнего ключа *НомерЗаказа* в таблицу *СтрокаЗаказа* из таблицы *Заказ* выполните следующее:
• Откройте таблицу *СтрокаЗаказа* в режиме *Конструктор*. Выделите поле *НомерЗаказа*.
• Выполните щелчок по вкладке *Подстановка* в *Свойствах поля*.
• Установите свойства подстановки в соответствии с рис. 26.

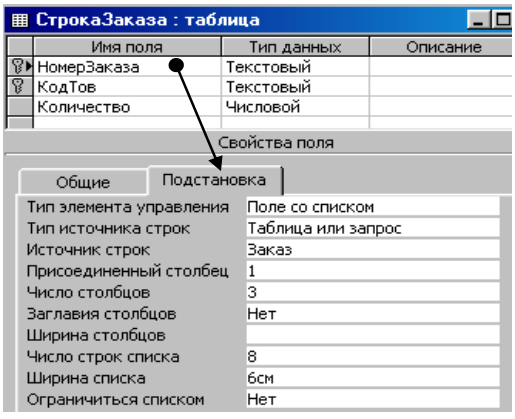


Рис. 26. **Свойства подстановки поля *НомерЗаказа* из таблицы *Заказ* в таблицу *СтрокаЗаказа***

Конец задания.

3.4. Заполнение базы данных

Заполнение таблиц БД фактическими данными обычно осуществляется с помощью форм. Формы для нашей БД создаются в подразделе *3.4.1 Создание форм*. В случае недостатка времени (например, из-за работы с версией Access, отличной от версии 2000) подраздел 3.4.1 может быть пропущен. В этом случае заполнение таблиц данными будет производиться в режиме *Таблица*.

3.4.1. Создание форм

Одной из основных целей создания форм является обеспечение удобного ввода данных в таблицы БД. Считается, что пользователю работать с формами удобнее, так как они имеют сходство (имитируют) с реальными документами. Кроме того, формы позволяют организовать входной контроль данных, производить корректировку данных и т. д. В СУБД Access формы могут быть созданы в режиме *Конструктор* с использованием *Мастера форм* и *Автоформы*.

Для лучшего понимания темы следует создать три формы: две однотабличные формы для таблиц *Клиент* и *Товар*, предназначенные для ввода нормативно-справочной информации о клиентах и товарах. Они будут созданы с помощью *Автоформы ленточная*, в которой поля, образующие одну запись, будут располагаться в одной строке.

Третья форма представляет собой многотабличную (двухтабличную) иерархическую форму. Форма для таблицы *Заказ* будет содержать подчиненную форму для таблицы *СтрокаЗаказа*. Эта иерархическая форма имитирует бланк заказа и предназначена для ввода оперативно-учетной информации о выполненных заказах. Она будет создана с помощью *Мастера форм*.

Задание 10. Создайте формы для таблиц *Клиент* и *Товар*. Для этого выполните следующее:

1. В окне базы данных выберите вкладку *Формы*, нажмите на кнопку *Создать*.
2. В диалоговом окне *Новая форма* выберите пункт *Автоформа: ленточная*.
3. В этом же окне в раскрывающемся списке источников данных выберите таблицу *Клиент* (*Товар*), нажмите на кнопку *ОК*.

Примечание. Автоформы создаются мгновенно. Каждая запись, включающая все поля таблицы *Клиент* (*Товар*), отображается в одну строку. В нижней части автоформы имеется набор кнопок со стрелками, позволяющими перемещаться по записям.

4. Сохраните форму под именем *Клиент* (*Товар*).

Примечание. Обратите внимание, что формам присвоены имена, совпадающие с именами таблиц, для просмотра, заполнения и корректировки которых и предназначены эти формы.

Конец задания.

Задание 11. Создайте форму для таблицы *Заказ* вместе с подчиненной формой для таблицы *СтрокаЗаказа*. Для этого выполните следующее:

1. Вызовите *Мастера форм* двойным щелчком по кнопке *Создание формы с помощью Мастера* во вкладке *Формы* окна базы данных.
2. В первом диалоговом окне *Создание форм Мастера форм* в раскрывающемся списке *Таблицы и запросы* выберите таблицу *Заказ*. Включите в форму все поля выбранной таблицы с помощью кнопки >>. Затем в этом же окне выберите таблицу *СтрокаЗаказа* и включите в форму все ее поля.
3. Во втором диалоговом окне *Мастера форм* выберите вид представления данных – *Заказ* и установите переключатель *Подчиненные формы*.
4. В третьем диалоговом окне *Мастера форм* выберите внешний вид подчиненной формы, установив переключатель *ленточный*.
5. В четвертом диалоговом окне *Мастера форм* выберите требуемый стиль – *Стандартный*.
6. В пятом диалоговом окне *Мастера форм* задайте следующие имена: для формы – *Заказ*, для подчиненной формы – *СтрокаЗаказа подчиненная форма*. Установите переключатель *Дальнейшие действия* в положение *Открыть форму для просмотра и ввода данных*. Нажмите на кнопку *Готово*.

Полученная форма представлена на рис. 27.

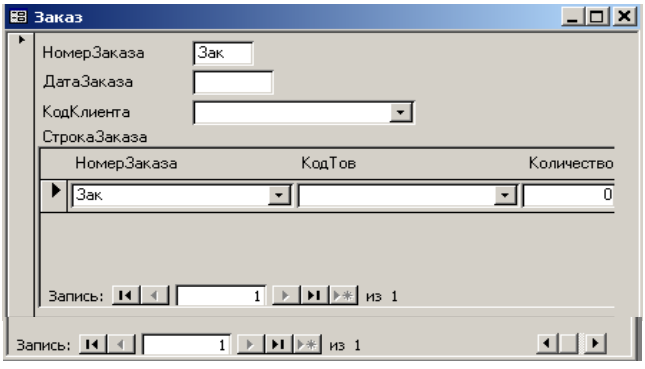


Рис. 27. **Форма *Заказ* вместе с подчиненной формой *СтрокаЗаказа***

Примечание. В Access 2000 форму для таблицы *Заказ* с подчиненной формой для таблицы *СтрокаЗаказа* можно получить также следующим образом: откройте таблицу *Заказ* (двойной щелчок по имени таблицы) и щелкните по кнопке *Новый объект: автоформы* на панели инструментов.

Конец задания.

3.4.2. Ввод данных в таблицы

Ввод данных в таблицы следует производить в следующей последовательности. Вначале заполняются таблицы с нормативно-справочной информацией (*Товар, Клиент*), затем – с оперативно-учетной информацией (*Заказ, СтрокаЗаказа*). При таком порядке заполнения таблиц можно использовать автоматическую подстановку значений внешнего ключа *КодКлиента* в таблицу *Заказ* из таблицы *Клиент*, а также значений внешнего ключа *КодТов* в таблицу *СтрокаЗаказа* из таблицы *Товар.*

Как было сказано ранее, ввод данных следует осуществлять или с помощью форм, созданных в подразделе 3.4.1, или непосредственно в режиме *Таблица*. Для ввода данных в таблицу необходимо открыть эту таблицу или одноименную форму. Открытие таблицы (формы) производится щелчком по кнопке *Открыть* во вкладке *Таблицы* (*Формы*) окна базы данных.

Ввод данных в таблицы *Товар* и *Клиент* с использованием созданных форм и в режиме *Таблица* ничем практически не отличаются.

Задание 12. Введите данные, приведенные в табл. 16, в таблицу *Клиент* базы данных *УчетЗаказов* с использованием созданной формы или непосредственно в режиме *Таблица.*

Таблица 16. **Данные для таблицы *Клиент***

КодКлиента	*НаимКлиент*	*КонтактЛицо*	*Адрес*
1	ОАО «Гомельские баранки»	Кашин	Гомельская, 16
2	ЗАО «Бубличная артель»	Козлевич	Одесская, 1
3	ЧУП «Нимфа»	Безенчук	Васюки, 2
4	ООО «Наше Бистро»	Казаков	Минская, 5
5	ЧТУП «Пьер и К»	Прошин	Брестская, 13

Конец задания.

Задание 13. Введите данные, приведенные в табл. 17, в таблицу *Товар* базы данных *УчетЗаказов* с использованием созданной формы или непосредственно в режиме *Таблица.*

Таблица 17. **Данные для таблицы *Товар***

КодТов	*НаимТовар*	*ЕдИзм*	*Цена*	*ИсхКолич*
АС	Абрикосы сушеные	кг	7760	56
БС	Бананы сушеные	кг	6570	87
ЖА	Желе апельсиновое	кг	4260	46
ЖВин	Желе виноградное	кг	3750	64
ЖВиш	Желе вишневое	кг	4500	76
ЖЛ	Желе лимонное	кг	3140	12
К	Кофе	кг	5760	56
ПС	Персики сушеные	кг	10870	68
СВин	Сидр виноградный	л	2790	97
СВиш	Сидр вишневый	л	3450	78
СЯ	Сидр яблочный	л	2150	102
ФС	Фиги сушеные	кг	8980	56

Конец задания.

Ввод информации в таблицы *Заказ* и *СтрокаЗаказа* в режиме *Таблица* отличается от ввода информации с использованием формы, имитирующей документ-бланк заказа (см. рис. 27).

В режиме *Таблица* вначале вводится информация в таблицу *Заказ* для шапки документа, а затем – в таблицу *СтрокаЗаказа* для табличной части. При таком порядке заполнения таблиц можно использовать автоматическую подстановку значений внешнего ключа *НомерЗаказа* в таблицу *СтрокаЗаказа* из таблицы *Заказ.*

При вводе информации с использованием формы (рис. 27), данные в каждом заказе вводятся сразу в обе таблицы. В этом случае автоматическая подстановка значений внешнего ключа *НомерЗаказа* в таблицу *СтрокаЗаказа* из таблицы *Заказ* реализуется в рамках одного бланка заказа: при вводе номера заказа в шапку бланка заказа он автоматически повторяется в строке (в строках) табличной части бланка заказа.

Задание 14. Введите данные, приведенные в таблицах 18–23, в таблицы *Заказ* и *СтрокаЗаказа* базы данных *УчетЗаказов* с использованием созданной формы или непосредственно в режиме *Таблица*.

Примечание. Обратите внимание на то, что вместо указанного в таблицах 18–23 наименования клиента в таблицу *Заказ* надо вводить с помощью подстановки соответствующий *КодКлиента*. Аналогично, вместо указанного в таблицах 18–23 наименования товара в таблицу *СтрокаЗаказа* надо вводить с помощью подстановки соответствующий *КодТов*.

Таблица 18. **Заказ № 1 от 02.12.2009 г.**
Клиент – ОАО «Гомельские баранки»

Наименование товара	Количество
Сидр яблочный	4
Желе апельсиновое	3

Таблица 19. **Заказ № 2 от 03.12.2009 г.**
Клиент – ЧУП «Нимфа»

Наименование товара	Количество
Сидр виноградный	12

Таблица 20. **Заказ № 3 от 03.12.2009 г.**
Клиент – ОАО «Гомельские баранки»

Наименование товара	Количество
Кофе	6
Сидр виноградный	4
Бананы сушеные	3

Таблица 21. **Заказ № 4 от 04.12.2009 г.**
Клиент – ОАО «Гомельские баранки»

Наименование товара	Количество
Желе вишневое	12
Желе апельсиновое	4

Таблица 22. **Заказ № 5 от 04.12.2009 г.**
Клиент – ЧТУП «Пьер и К»

Наименование товара	Количество
Желе вишневое	13
Желе виноградное	34
Желе апельсиновое	4

Таблица 23. **Заказ № 6 от 05.12.2009 г.**
Клиент – ЧУП «Нимфа»

Наименование товара	Количество
Желе вишневое	10
Кофе	12

Конец задания.

3.5. Использование БД

После заполнения БД оперативно-учетной информацией (и параллельно с заполнением) начинается этап использования (эксплуатации) БД – формирование запросов и отчетов.

3.5.1. Создание запросов

Среди многочисленных функций СУБД главная – возможность выдавать ответы на поступающие запросы. Дело в том, что обычно базы данных содержат огромный объем информации, который пользователь не может в целом «переварить» и осмыслить. Он только может работать с отдельными «срезами» информации и извлекать их из БД с помощью запросов.

Классификация запросов в СУБД Access приведена на рис. 28. Они разделяются на две группы: *запросы на выборку* и *активные запросы*.

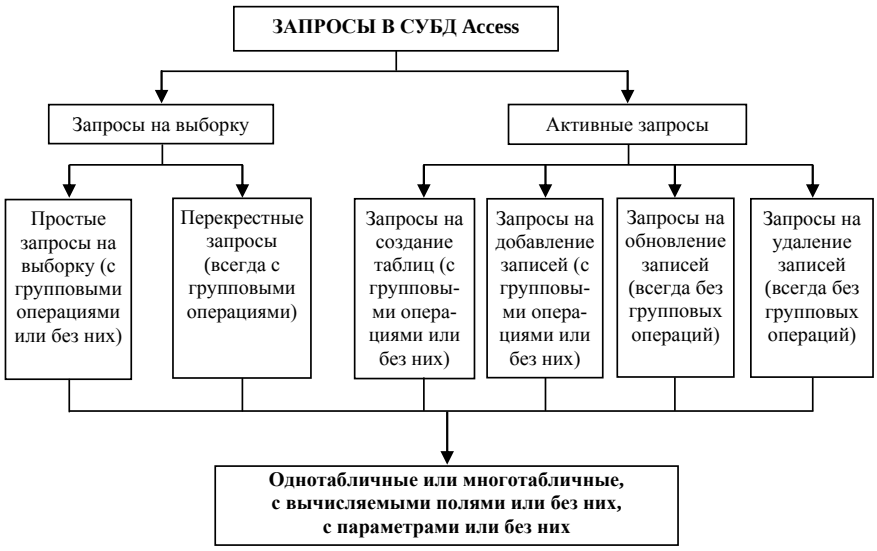


Рис. 28. **Классификация запросов в СУБД Access**

Основное отличие двух групп запросов заключается в следующем: запросы на выборку извлекают информацию из БД, не изменяя БД, а активные запросы изменяют существующую БД (модифицируют информацию в базовых таблицах, добавляют и удаляют таблицы и т. д.).

В данном пособии будут создаваться только запросы на выборку.

Запросы обеих групп могут быть *однотабличными* и *многотабличными*, т. е. для их реализации в качестве исходных данных используется информация, содержащаяся в одной или многих таблицах (в том числе таблицах, полученных как результат другого запроса).

В каждом запросе могут быть *вычисляемые поля*, т. е. их значения формируются из значений других полей посредством формул.

Некоторые виды запросов могут предусматривать выполнение групповых операций, т. е. вычислений с использованием данных из некоторой группы записей.

Среди *запросов на выборку* можно выделить простые запросы на выборку и перекрестные запросы.

Простой запрос на выборку содержит формулировку запроса к базе данных, определяемую как набор критериев для выборки интересующих данных из одной или более таблиц.

Для создания запроса целесообразно использовать *Конструктор запросов*.

Окно *Конструктор запросов* состоит из двух частей, разделенных горизонтальной линией, в верхней части которого располагаются таблицы, выбранные ранее в диалоговом окне *Добавление таблицы*. Если запрос формируется на основе только таблиц, входящих в базу данных *УчетЗаказов*, то между этими таблицами автоматически устанавливаются связи такие же, как и в схеме данных, приведенной на рис. 23.

Примечание. Если добавлялись таблицы, представляющие собой результаты других запросов, то для таких таблиц необходимо устанавливать связи с другими таблицами. Обычно устанавливается связь, соответствующая операции экви-соединения реляционной алгебры [1], которая в Access названа внутренним объединением.

В нижней части окна *Конструктор запросов* располагается *бланк запроса*, который часто называют *бланком QBE*. С помощью этого бланка, собственно, и происходит конструирование запроса.

На бланк *QBE* переносятся из верхней части окна *Конструктор запросов* поля из исходных таблиц (при этом автоматически заполняются строки бланка *Поле* и *Имя таблицы*), затем формируются критерии отбора записей (в строке *Условие отбора*) и выполняются другие необходимые операции (например, устанавливается флажок в строке *Вывод на экран*).

Перенос полей на бланк *QBE* осуществляется буксировкой этих полей (*drag&drop*) или двойным щелчком по полю в исходной таблице запроса.

Бланк *QBE* по умолчанию состоит из строк *Поле*, *Имя таблицы*, *Сортировка*, *Вывод на экран*, *Условие отбора*, *или*.

В ячейки строки *Поле* вводятся имена полей запроса. Имя поля запроса представляет собой или имя поля исходной таблицы запроса, или имя нового вычисляемого поля, в этом случае оно отделяется двоеточием от вычисляемого выражения.

В ячейки строки *Имя таблицы* вводятся имена исходных таблиц запроса. Для вычисляемых полей имя таблицы не указывается.

В ячейках строки *Сортировка* имеется раскрывающийся список, позволяющий выбрать порядок сортировки записей результата запроса по значениям поля (*по возрастанию*, *по убыванию*, *сортировка отсутствует*).

В строке *Вывод на экран* отмечаются флажком поля, которые должны быть видны на экране после выполнения запроса.

В ячейках строк *Условие отбора* и *или* указываются условия отбора записей в виде произвольных логических выражений. Если условия, накладываемые на значения из различных полей, связаны между собой логическим оператором *И*, то эти условия указываются на одной строке, если же они связаны между собой логическим оператором *ИЛИ*, то эти условия указываются на разных строках.

Строка *Групповая операция* появляется на *бланке QBE* по команде *Вид | Групповые операции* (или после нажатия на кнопку с изображением знака суммы на панели инструментов).

Групповые операции определяют группы записей, для отдельных полей которых выполняются вычисления статистических функций, выбираемых из раскрывающегося списка:

- *Sum* – сумма значений поля в группе записей;
- *Avg* – среднее от значений поля в группе записей;
- *Min* – наименьшее значение поля в группе записей;
- *Max* – наибольшее значение поля в группе записей;
- *Count* – число значений поля без учета пустых значений в группе записей;
- *StDev* – среднеквадратичное отклонение от среднего значения поля в группе записей.

Задание 15. Создайте запрос *ДорогаяПятерка* на выборку пяти наименований товаров с наибольшей ценой за единицу (килограмм или литр).

Примечание. Запрос *ДорогаяПятерка* должен быть однотабличным, так как исходные данные находятся в одной таблице *Товар*.

Порядок создания запроса следующий:

1. В окне базы данных выберите вкладку *Запросы* и нажмите на кнопку *Создать*.
2. В диалоговом окне *Новый запрос* выберите пункт *Конструктор*, нажмите на кнопку *ОК*. Появятся окно *Конструктора запросов* и окно *Добавление таблицы*.
3. В окне *Добавление таблицы* выделите таблицу *Товар* и нажмите на кнопку *Добавить*. Закройте окно *Добавление таблицы*. На экране останется окно *Конструктора запросов* с именем *Запрос1: запрос на выборку*.
4. В ячейки строки *Поле* бланка *QBE* добавьте поля *НаимТовар* и *Цена*.
5. Установите сортировку записей *по убыванию* значений поля *Цена*.
6. На панели инструментов в раскрывающемся списке *Набор значений* выберите значение 5. Перейдите в представление запроса *Режим таблицы*. Результат выполнения задания приведен на рис. 29.

Дорогая Пятерка : запрос на выборку

НаимТовар	Цена
Персики сушеные	10870
Фиги сушеные	8980
Абрикосы сушеные	7760
Бананы сушеные	6570
Кофе	5760

Рис. 29. **Результат запроса *ДорогаяПятерка***

7. Закройте окно запроса (с его сохранением под именем *ДорогаяПятерка*).

Конец задания.

Задание 16. Создайте запрос *НимфаТовары* на выборку наименований товаров, купленных (заказанных) фирмой *Нимфа*.

Искомый запрос в режиме *Конструктор* представлен на рис. 30.

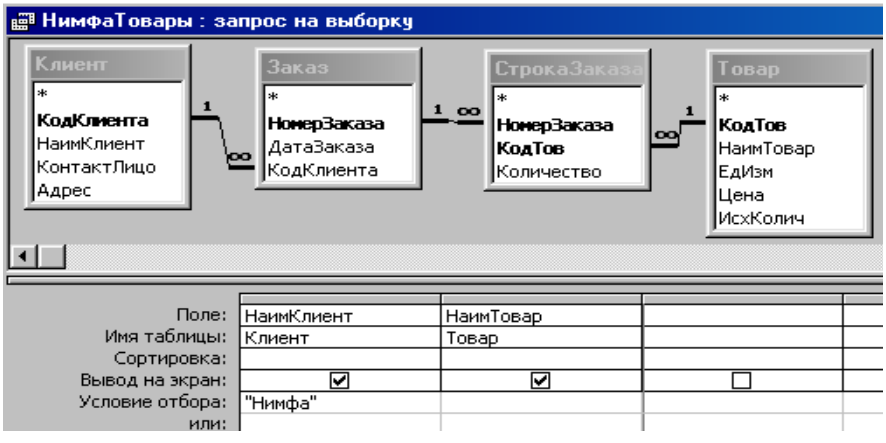


Рис. 30. **Запрос *НимфаТовары* в режиме *Конструктор***

Результат выполнения искомого запроса представлен на рис. 31.

НимфаТовары : запрос на выборку

НаимКлиент	НаимТовар
Нимфа	Сидр виноградный
Нимфа	Желе вишневое
Нимфа	Кофе

Рис. 31. **Результат выполнения запроса *НимфаТовары***

Конец задания.

Задание 17. Создайте запрос *НимфаИлиПьерТовары* на выборку наименований товаров, купленных фирмой *Нимфа* или фирмой *Пьер и К*.

Искомый запрос в режиме конструктора представлен на рис. 32.

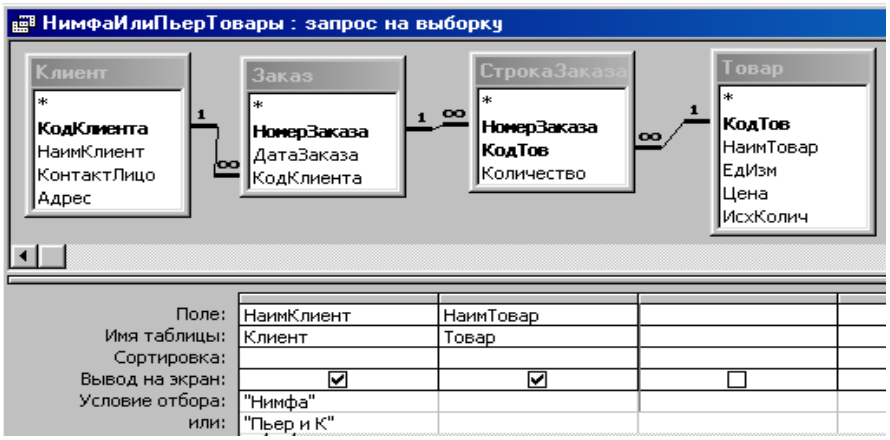


Рис. 32. **Запрос *НимфаИлиПьерТовары* в режиме *Конструктор***

Результат выполнения искомого запроса представлен на рис. 33.

НимфаИлиПьерТовары : запрос на вы

НаимКлиент	НаимТовар
Нимфа	Сидр виноградный
Нимфа	Желе вишневое
Нимфа	Кофе
Пьер и К	Желе вишневое
Пьер и К	Желе виноградное
Пьер и К	Желе апельсиновое

Рис. 33. **Результат выполнения запроса *НимфаИлиПьерТовары***

Конец задания.

Задание 18. Создайте запрос *НимфаСо2По4Товары* на выборку наименований товаров, купленных ЧТУП «Нимфа» со 2 по 4 декабря 2009 г.

Искомый запрос в режиме *Конструктор* представлен на рис. 34.

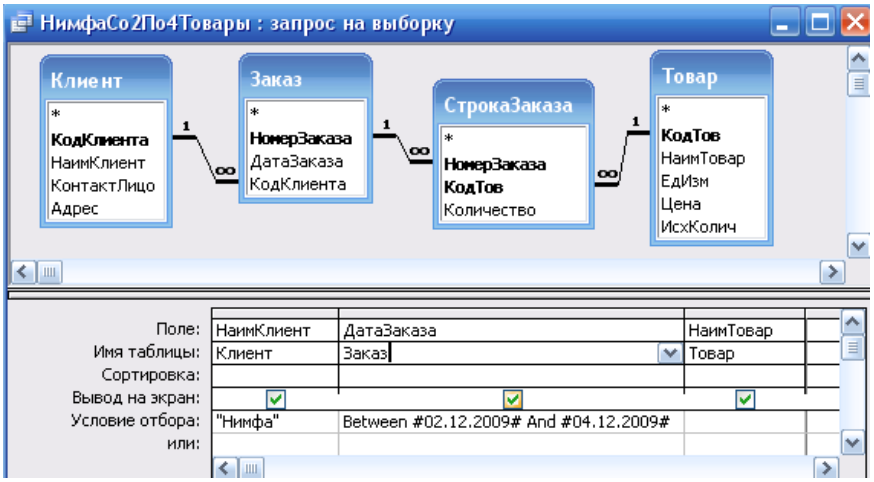


Рис. 34. **Запрос *НимфаСо2По4Товары* в режиме *Конструктор***

Результат выполнения искомого запроса представлен на рис. 35.

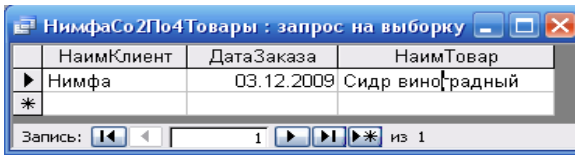


Рис. 35. **Результат выполнения запроса *НимфаСо2По4Товары***

Конец задания.

Задание 19. Создайте запрос *КлиентыСуммыПострочно* на выборку для всех клиентов наименований купленных ими товаров с указанием для всех заказов даты заказа и суммы по каждой строке заказа.

Запрос *КлиентыСуммыПострочно* содержит *вычисляемое* поле *Сумма*. Значение в этом поле получается путем умножения значения поля *Цена* таблицы *Товар* на значение поля *Количество* таблицы *СтрокаЗаказа*.

Для создания вычисляемого поля *Сумма* выполните следующее:

1. Щелкните мышкой по свободной ячейке строки *Поле* и нажмите на кнопку *Построить* на панели инструментов. Появится окно *Построитель выражений* (рис. 36). Требуемое выражение создается в поле построителя в верхней половине окна. Для того, чтобы вставить элемент в поле *Построителя выражений*, выделите этот элемент в нижней половине окна построителя и нажмите кнопку *Вставить* (или выполните двойной щелчок мышью по этому элементу).

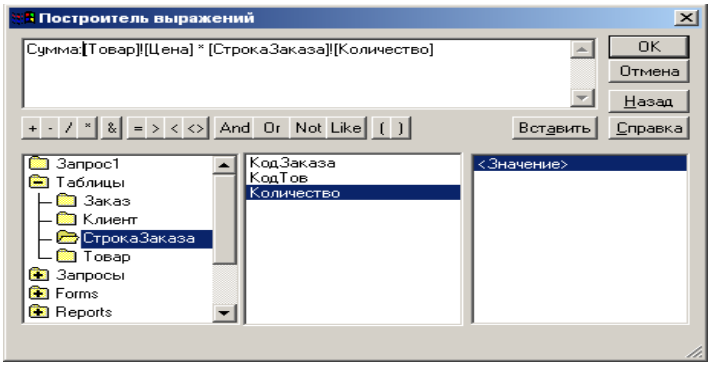


Рис. 36. **Окно *Построитель выражений***

2. Раскройте состав таблиц базы данных двойным щелчком по папке *Таблицы*, которая находится в левом поле нижней половины окна *Построителя выражений*.

3. Выделите таблицу *Товар*. В среднем поле нижней половины окна *Построителя выражений* появится список полей таблицы *Товар*.

4. Выделите поле *Цена* и нажмите на кнопку *Вставить*. Вставляемое в выражение имя поля предваряется именем таблицы.

5. Вставьте в выражение знак умножения и поле *Количество* из таблицы *СтрокаЗаказа*.

6. Перед выражением введите имя поля *Сумма*. Между именем поля и выражением ставится двоеточие. Пробелы не вводятся. Нажмите на кнопку *ОК*.

7. Завершите ввод выражения в ячейку бланка запроса щелчком в любом другом месте бланка запроса.

Искомый запрос в режиме *Конструктор* представлен на рис. 37.

Рис. 37. **Запрос *КлиентыСуммыПострочно* в режиме *Конструктор***

Результат выполнения искомого запроса представлен на рис. 38.

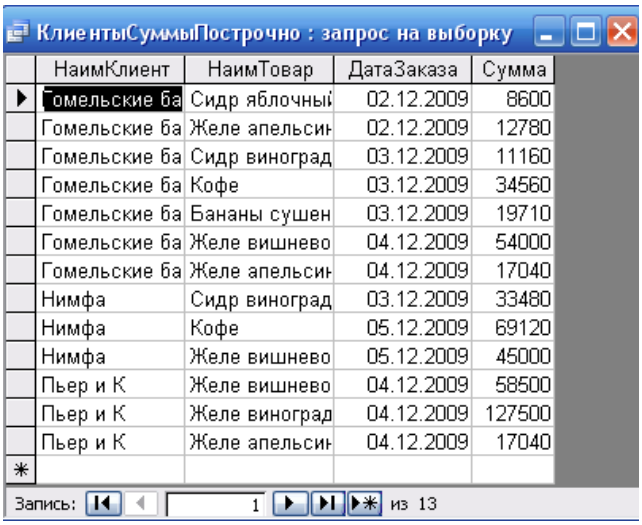


Рис. 38. **Результат выполнения запроса *КлиентыСуммыПострочно***

Конец задания.

Задание 20. Создайте запрос *КлиентыСуммыПодатам* на выборку для каждого клиента для каждой даты покупки суммы всех купленных товаров.

Запрос *КлиентыСуммыПодатам* создается на основе предыдущего запроса *КлиентыСуммыПострочно*. В этом запросе используются групповые операции. Для групп записей, относящихся к заказам конкретных клиентов за конкретную дату, производится вычисление суммы (используется функция *Sum*) значений поля *Сумма* из таблицы-результата предыдущего запроса *КлиентыСуммыПострочно*.

Искомый запрос в режиме *Конструктор* представлен на рис. 39.

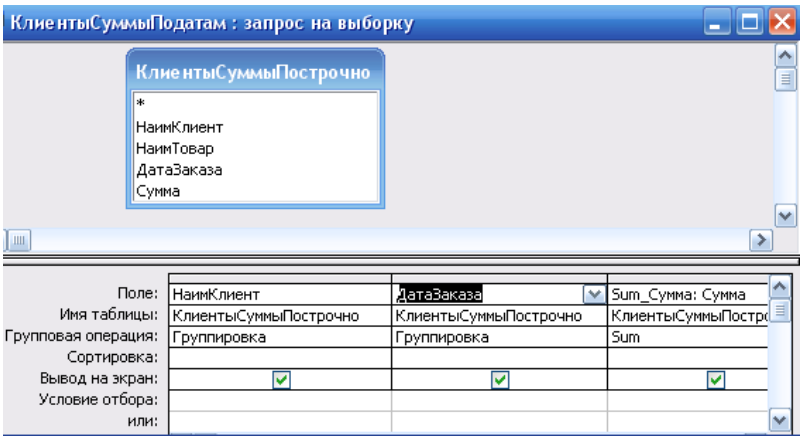


Рис. 39. **Запрос *КлиентыСуммыПодатам* в режиме *Конструктор***

На рис. 40 представлен результат выполнения искомого запроса.

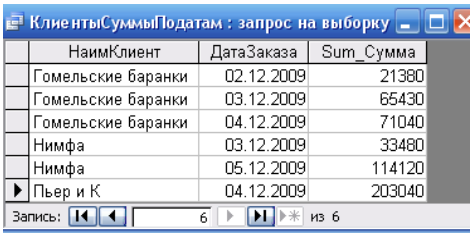


Рис. 40. **Результат выполнения запроса *КлиентыСуммыПодатам***

Конец задания.

3.5.2. Формирование отчета о продажах за анализируемый период

Задание 21. Создайте отчет о продажах за анализируемый период (02.12.2009 г. – 05.12.2009 г.).

Создавать отчет можно с помощью *Мастера отчетов*. В качестве источника данных для отчета выберем запрос *КлиентыСуммыПодатам*. Порядок создания отчета следующий:

1. В окне базы данных выберите вкладку *Отчеты*, нажмите на кнопку *Создать*.
2. В диалоговом окне *Новый отчет* выделите пункт *Мастер отчетов* и выберите в качестве источника данных запрос *КлиентыСуммыПодатам*. Нажмите на кнопку *ОК*.
3. В окне *Создание отчетов* выберите для отчета все поля запроса *КлиентыСуммыПодатам* с помощью кнопки >>.
4. На следующем шаге (переход к следующему шагу осуществляется по командной кнопке *Далее* >) *Мастера отчетов* выполняется группировка. Выделите поле *ДатаЗаказа* и нажмите на кнопку со стрелкой (вправо). Если по умолчанию установлен интервал группировки не по дням, то нажмите на командную кнопку *Группировка…* и в раскрывающемся списке *Интервалы группировки* выберите *по дням*.
5. Далее задается сортировка записей. Сортировка может производиться одновременно по нескольким (до четырех) полям. Выберите в первом раскрывающемся списке поле *НаимКлиент*, во втором – *Sum_Сумма*. Записи сортируйте по возрастанию (от А до Я) значений поля *Клиент* и по убыванию (от Я до А) значений поля *Sum_Сумма*. Затем нажмите на командную кнопку *Итоги...* и в окне *Итоги* выберите операцию *Sum* для поля *Sum_Сумма*. Переключатель *Показать* установите в положение *Данные и итоги*, нажмите на кнопку *ОК*, перейдите к следующему шагу.
6. Выберите для отчета макет *Блок*, ориентацию *Альбомная*, перейдите к следующему шагу.
7. Выберите для отчета стиль *Обычный*.
8. На последнем шаге *Мастера отчетов* задайте имя отчета *Отчет о продажах за декабрь*, установите переключатель *Дальнейшие действия:* в положение *Просмотреть отчет* и нажмите на кнопку *Готово*.

Сформированный отчет приведен на рис. 41.

Отчет о продажах за декабрь

ДатаЗаказа по дням	Клиент	Sum_Сумм	ДатаЗаказа
2 Декабрь 2009 г.	Гомельские баранки	21380	02.12.2009
Итоги для 'ДатаЗаказа' = 02.12.2009 (1 запись)			
Sum		**21380**	
3 Декабрь 2009 г.	Гомельские баранки	65430	03.12.2009
	Нимфа	33480	03.12.2009
Итоги для 'ДатаЗаказа' = 03.12.2009 (2 записей)			
Sum		**98910**	
4 Декабрь 2009 г.	Гомельские баранки	71040	04.12.2009
	Пьер и К	203040	04.12.2009
Итоги для 'ДатаЗаказа' = 04.12.2009 (2 записей)			
Sum		**274080**	
5 Декабрь 2009 г.	Нимфа	114120	05.12.2009
Итоги для 'ДатаЗаказа' = 05.12.2009 (1 запись)			
Sum		**114120**	
ИТОГО		**508490**	

Рис. 41. **Отчет, сформированный *Мастером отчетов***

Конец задания.

4. МЕЖТАБЛИЧНЫЕ СВЯЗИ И КОНСТРУИРОВАНИЕ ЗАПРОСОВ В СУБД ACCESS

4.1. Межтабличные связи в СУБД Access

Реляционная база данных, состоящая из многих таблиц, имеет, как правило, схему данных, в которой указываются связи между таблицами. В СУБД MS Access между двумя таблицами могут быть установлены связи двух видов: *связи целостности данных* (их еще называют *связями ссылочной целостности*) и *связи-соединения*.

Примечание. В MS Access связи-соединения называются связями-объединениями. Дело в том, что при русскоязычной локализации MS Access была допущена досадная ошибка. Вместо общепринятых на русском языке названий операций над отношениями *СОЕДИНЕНИЕ* – для *JOIN* и *ОБЪЕДИНЕНИЕ* – для *UNION* оба были переведены как *ОБЪЕДИНЕНИЕ*.

При определении этих связей необходимо различать таблицы двух видов: таблицы, составляющие собственно базу данных, и временные таблицы, представляющие собой результаты запросов. Их иногда называют, соответственно, базовыми таблицами и просто запросами.

Оба вида связи устанавливаются по полям таблиц, имеющим одинаковый или совместимый тип данных. Связи целостности данных могут быть установлены только между базовыми таблицами из одной базы данных. Связи-соединения могут быть установлены между любыми таблицами.

Для установления связи целостности данных необходимо, чтобы связываемые поля были первичным ключом хотя бы в одной из связываемых базовых таблиц. Если связываемые поля являются первичным ключом или имеют неповторяющиеся значения в обеих базовых таблицах, то устанавливается связь «*один к одному*», иначе – «*один ко многим*». Последний тип связи целостности данных встречается наиболее часто.

Связи целостности данных создаются путем установки флажка *Обеспечение целостности данных* в диалоговом окне *Изменение связей*, которое вызывается по команде *Сервис\ Схема данных\ Связи\ Изменить связь*.

Установка дополнительных флажков *каскадное обновление связанных полей* и *каскадное удаление связанных полей* только уточняет порядок обеспечения целостности данных.

Если флажок *каскадное обновление связанных полей* не установлен, то Access не позволит изменять значение ключевого поля в главной таблице, для которого имеются связанные записи в подчиненной таблице.

Если флажок *каскадное удаление связанных полей* не установлен, то Access не позволит удалять записи из главной таблицы, для которых имеются связанные записи в подчиненной таблице.

Одновременно с установлением связи целостности данных между базовыми таблицами сразу же устанавливается и один из трех способов связи-соединения. Другими словами, связь целостности данных «нагружается» еще и связью-соединением.

Связи-соединения между таблицами могут также устанавливаться при формировании запроса в режиме *Конструктор*. Собственно при реализации запросов и используются связи-соединения.

Как было сказано выше, связи-соединения могут устанавливаться между двумя базовыми таблицами, между двумя запросами и между базовой таблицей и запросом.

Поэтому главное, чтобы поля связи у обеих таблиц имели одинаковый или совместимый тип данных.

Связи-соединения определяют, каким образом из записей связываемых таблиц формируется соединенная таблица. Возможны три способа формирования соединенной таблицы:

1. Соединение только тех записей, в которых связанные поля обеих таблиц совпадают (аналог экви-соединения реляционной алгебры).
2. Соединение всех записей из первой таблицы и только тех записей из второй таблицы, в которых связанные поля совпадают (аналог левого внешнего соединения реляционной алгебры).
3. Соединение всех записей из второй таблицы и только тех записей из первой таблицы, в которых связанные поля совпадают (аналог правого внешнего соединения реляционной алгебры).

Примечание. Возможны также соединения, в которые записи попадают не только по равенству значений соединяемых полей, но и по результатам других операций сравнения (>, <, >=, <=, <>). Это так называемые тэта-соединения, которые в данном пособии не рассматриваются. Такие соединения на *QBE* могут быть реализованы (без явного (графического) указания связей) применением операции выборки к декартову произведению соединяемых таблиц.

Способы соединения устанавливаются в диалоговом окне *Параметры объединения* (конечно, точнее было бы *Параметры соединения*), вызываемом из диалогового окна *Изменение связей* по командной кнопке *Объединение*… .

При работе с базовыми таблицами в режиме *Таблица* определенное значение имеет только флажок *Обеспечение целостности данных* и совершенно безразличны способы их соединения, т. е. целостность данных важна только при наполнении БД фактическими данными. Способы соединения играют роль только при конструировании многотабличных запросов, так как определяют, каким образом из записей используемых в запросе таблиц формируется соединенная исходная таблица запроса (одна таблица). При этом не важно, каким образом получено содержимое исходных (базовых) таблиц запроса, с соблюдением целостности данных или с нарушением.

В качестве иллюстрации к вышесказанному на рисунках 42–45 показано, как выглядят в *Конструкторе запросов* СУБД Access «чистые» связи экви-соединения и левого внешнего соединения, а также связи, нагруженные требованием соблюдения ссылочной целостности.

Рис. 42. **Связь экви-соединения таблиц *Поставщик* и *Поставки***

Рис. 43. **Связь левого внешнего соединения таблиц *Поставщик* и *Поставки***

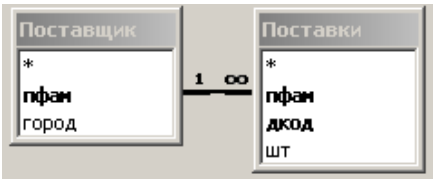


Рис. 44. **Связь экви-соединения таблиц *Поставщик* и *Поставки*, нагруженная связью ссылочной целостности**

Рис. 45. **Связь левого внешнего соединения таблиц *Поставщик* и *Поставки*, нагруженная связью ссылочной целостности**

4.2. Конструирование запросов в СУБД Access

Пользоваться при создании запросов справочной системой Access достаточно сложно. Несмотря на то, что в соответствующих разделах справочной системы описана реализация каждого конкретного вида запроса, отсутствует общий методический прием конструирования запросов различного вида. Нет четкого определения исходной информации для реализации запроса, не приведена простая в усвоении и логически стройная типовая последовательность операций над исходной информацией при построении запроса. Добиться всего этого можно, если использовать понятие соединенной исходной таблицы запроса (СИТЗ) и операции реляционной алгебры над ней.

Суть такого подхода при изучении запросов состоит в том, что определяются три следующих начальных шага при формировании всех видов запросов:

1. Реализуются все связи-соединения между исходными таблицами запроса, заведомо содержащими все требуемые для выполнения запроса данные, т. е. создается СИТЗ. Отметим, что в случае однотабличного запроса СИТЗ совпадает с данной исходной таблицей. Результатом выполнения первого шага является, по сути, сведение многотабличного запроса к однотабличному запросу над СИТЗ.

2. К СИТЗ применяется операция *Проекция* реляционной алгебры. При этом происходит «прореживание» СИТЗ по столбцам.

3. К полученной на втором шаге таблице применяется операция *Выборка* реляционной алгебры. При этом происходит ее «прореживание» по строкам.

Затем над полученной после этих трех шагов «прореженной» таблицей выполняются другие операции, которые зависят от вида запроса.

Для вызова *Конструктора запросов* необходимо в окне базы данных перейти на вкладку *Запросы* и выполнить двойной щелчок по кнопке *Создание запроса* в режиме *Конструктор*. В появившемся диалоговом окне *Добавление таблицы* Access предложит выбрать таблицы, содержащие информацию, необходимую для реализации запроса. После выбора этих таблиц появляется окно *Конструктор запросов*, в верхней части которого располагаются выбранные таблицы, в нижней – так называемый бланк *QBE*.

Между исходными базовыми таблицами запроса отображаются связи, ранее установленные в схеме данных БД. Кроме того, Access дополнительно автоматически устанавливает экви-соединения (внутренние соединения) для таблиц, у которых имеются поля с одинаковыми именами и совместимыми типами данных. При необходимости все эти связи можно откорректировать и дополнить вручную.

В итоге все исходные таблицы запроса вместе с их связями-соединениями определяют единую исходную таблицу, СИТЗ, из которой будет извлекаться информация для реализации запроса. Другими словами, исходной информацией для реализации запроса является таблица, полученная после неявного выполнения соответствующих операций соединения между всеми связанными таблицами. Отметим здесь, что в том случае, когда между исходными таблицами запроса нет связи-соединения, в качестве единой исходной таблицы берется декартово произведение этих несвязанных таблиц.

Затем на бланк *QBE* буксируются поля из исходных таблиц, формируются критерии отбора записей и выполняются другие необходимые операции. Предполагается, что поля, необходимые только для расчета значений в вычисляемых полях, также буксируются в бланк *QBE*.

Если в бланк *QBE* буксируются не все поля из исходных таблиц, то происходит, фактически, «прореживание» СИТЗ по столбцам. Таким образом, выбор полей в бланк *QBE* «почти» эквивалентен операции *Проекция* реляционной алгебры над СИТЗ. «Почти», так как Access не удаляет автоматически строки-дубликаты, которые могут появиться в результате.

Условия отбора, заносимые в соответствующие строки бланка QBE, определяют подмножество отбираемых строк СИТЗ, т. е. происходит, фактически, «прореживание» СИТЗ по строкам. Таким образом, формирование условий отбора в бланке *QBE* эквивалентно операции *Выборка* реляционной алгебры над СИТЗ.

В случае простого запроса на выборку, не имеющего параметров и не предусматривающего создание вычисляемых полей и использование групповых операций, «прореженная» по столбцам и строкам СИТЗ сразу дает результат выполнения запроса.

Если в запросе предполагается создание вычисляемых полей, то такие поля добавляются к «прореженной» СИТЗ и им присваиваются имена.

В случае использования в запросе групповых операций производится группировка записей «прореженной» СИТЗ и для некоторых полей выполняется вычисление одной из статистических функций над значениями этих полей в группах записей. Это такие функции, как *Sum, Min, Count* и др.

В случае перекрестного запроса одно или несколько полей, значения которых станут заголовками строк, поле (только одно), значения которого станут заголовками столбцов, а также поле обработки берутся из «прореженной» СИТЗ.

Пример 8. В качестве примера рассмотрим, как выполняются указанные три шага при реализации с помощью *Конструктора запросов* запроса к базе данных *Поставщики-Детали*:

1. Определить общее количество гаек, которое поставил Смит.

Для наглядности на рисунках 46 и 47 еще раз приведено содержимое таблиц базы данных и схема данных.

Очевидно, исходные данные, содержащие всю необходимую информацию для реализации запроса, имеются в таблицах *Деталь* и *Поставки*. После добавления этих таблиц в верхнюю часть окна *Конструктор запросов* между этими таблицами автоматически устанавливаются связи как на схеме данных БД *Поставщики-Детали*.

Деталь : таблица

	дкод	дназв	цвет	вес
⊞	д1	болт	черный	10
⊞	д2	гайка	черный	4
⊞	д3	гайка	красный	5
⊞	д4	винт	зеленый	7

Поставщик : таблица

	пфам	город
⊞	Смит	Лондон
⊞	Джонс	Париж
⊞	Грис	Лондон

Поставки : таблица

пфам	дкод	шт
Джонс	д1	9
Джонс	д2	4
Смит	д1	3
Смит	д2	7
Смит	д3	2

Рис. 46. **Содержимое таблиц база данных *Поставщики-Детали***

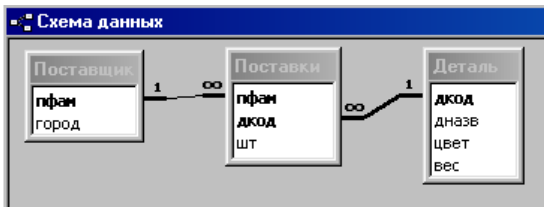


Рис. 47. **Схема данных БД *Поставщики-Детали***

Результат первого шага, т. е. создание СИТЗ, как экви-соединения таблиц *Деталь* и *Поставки*, представлен на рисунках 48 и 49 в режиме *Конструктор* и в режиме *Таблица*, соответственно.

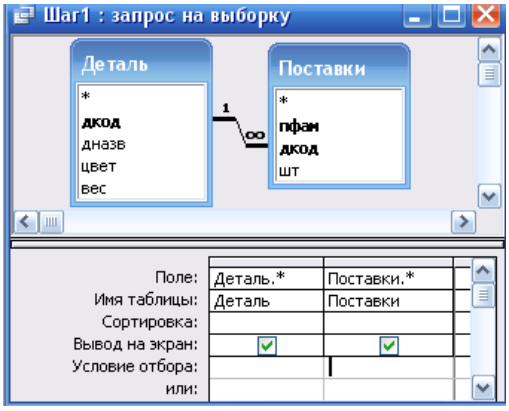


Рис. 48. **Реализация первого шага (создание СИТЗ) в режиме *Конструктор***

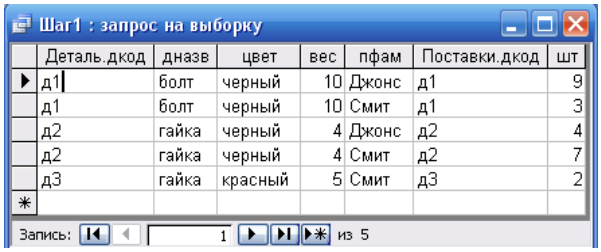


Рис. 49. **СИТЗ**

На втором шаге производится прореживание СИТЗ по столбцам. В данном случае из СИТЗ вырезаются три столбца из семи: *дназв*, *пфам* и *шт*. Реализация второго шага в режиме *Конструктор* и в режиме *Таблица* приведена на рисунках 50 и 51.

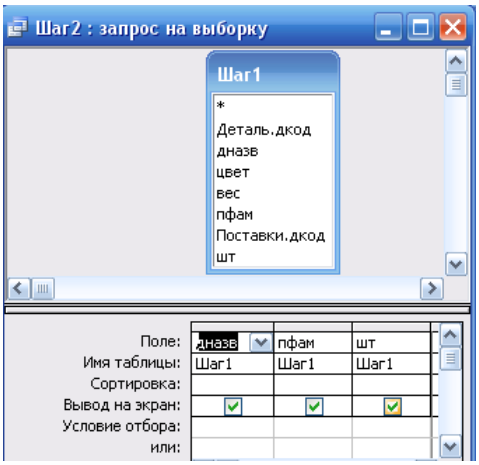


Рис. 50. **Реализация второго шага («прореживание» СИТЗ по столбцам) в режиме *Конструктор***

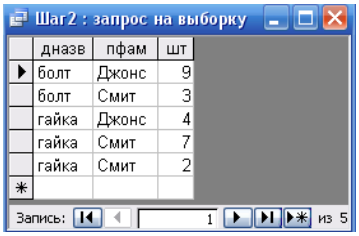


Рис. 51. **СИТЗ, «прореженная» по столбцам**

Следует отметить, что результат второго шага может не совпадать с результатом операции *Проекция* реляционной алгебры, например, в случае ее применения к СИТЗ по полям *дназв* и *пфам* (объясните, почему).

3. На третьем шаге результат второго шага прореживается по строкам по следующему условию: в поле *дназв* должно быть записано «гайка» *И* в поле *пфам Смит*. Это аналог операции *ВЫБОРКА* реляционной алгебры, примененной к результату второго шага по условию

(*дназв* = «гайка») *И* (*пфам* = «Смит»).

Реализация третьего шага в режиме *Конструктор* и в режиме таблицы приведена на рисунках 52 и 53.

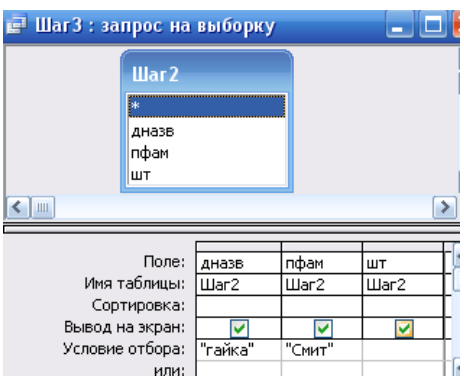


Рис. 52. **Реализация третьего шага («прореживание» СИТЗ по столбцам) в режиме *Конструктор***

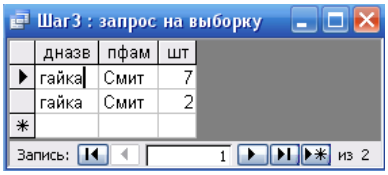


Рис. 53. **СИТЗ, «прореженная» и по столбцам, и по строкам**

Таким образом, исходными данными для реализации запроса *Сколько же гаек поставил Смит?* является таблица, полученная в результате выполнения третьего шага. К этой таблице применяется групповая операция, позволяющая вычислить статистическую функцию *Sum* по полю *шт* для всех строк таблицы. Окончательный результат запроса приведен на рисунках 54 и 55.

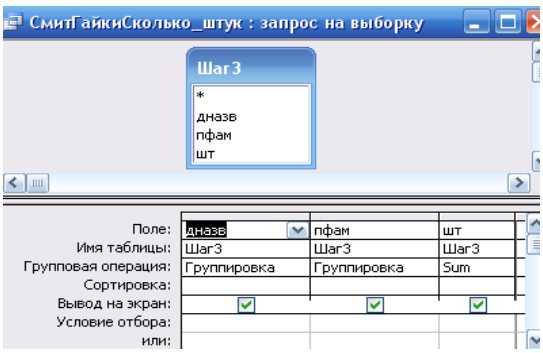


Рис. 54. **Реализация запроса *Сколько гаек поставил Смит?* на основе таблицы, полученной при выполнении третьего шага**

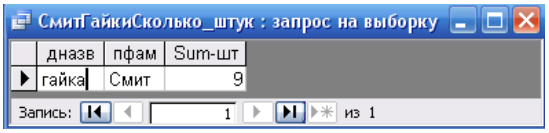


Рис. 55. **Результат запроса *Сколько гаек поставил Смит?***

Такова логическая последовательность выполнения запроса *Сколько гаек поставил Смит?*, приведенного на рис. 56.

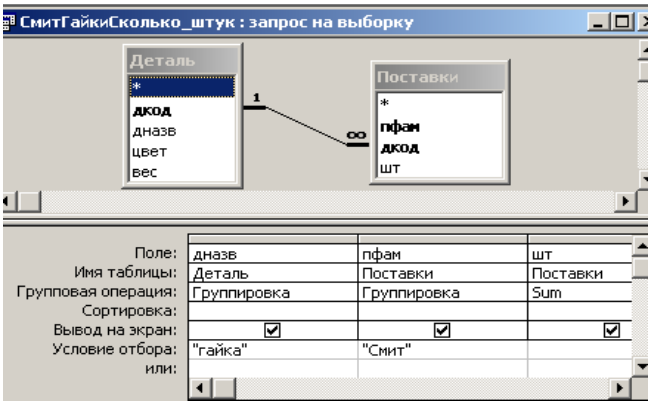


Рис. 56. **Окно *Конструктора запросов*, реализующего запрос *Сколько гаек поставил Смит?***

Следует подчеркнуть, что здесь имеется в виду логическая, но не фактическая физическая реализация запроса. Такое пошаговое представление выполнения запроса должно облегчить пользователю осмысленное применение инструментов *Конструктора запросов* при создании произвольных запросов к базе данных.

5. ВВЕДЕНИЕ В SQL

Как указывалось в подразделе 1.3, стандарт языка *SQL* слишком велик по объему, и ни одна СУБД не поддерживает его в полной мере. Кроме того, производители СУБД часто предлагают собственные расширения *SQL*, несовместимые друг с другом.

Поэтому здесь приводится только весьма краткое и урезанное описание возможностей языка *SQL*. Мы рассматриваем некое подмножество языка, которое дает общее представление о его специфике и возможностях и позволяет начать самостоятельную работу в среде выбранной СУБД.

SQL-DDL (*Data Definition Language*) – язык определения структур и ограничений целостности баз данных. К нему относятся команды создания и удаления баз данных, а также команды создания, изменения и удаления таблиц и т. д.

SQL-DML (*Data Manipulation Language*) – язык манипулирования данными путем добавления, изменения, удаления и извлечения данных, а также управление транзакциями.

В *SQL* нет таких формальных терминов, как «отношение», «кортеж», «атрибут». Вместо них используются «таблица», «строка», «столбец».

Все примеры построены применительно к базе данных *Учет заказов*.

5.1. Типы данных SQL

Рассмотрим основные типы данных языка *SQL*:

1. *Символьные типы данных* содержат буквы, цифры и специальные символы.

• *CHAR* или *CHAR*(*n*) – символьные строки фиксированной длины. Длина строки определяется параметром (*n*). *CHAR* без параметра соответствует *CHAR*(1). Для хранения таких данных всегда отводится *n* байт вне зависимости от реальной длины строки.

• *VARCHAR*(*n*) – символьная строка переменной длины. Для хранения данных этого типа отводится число байт, соответствующее реальной длине строки.

2. *Целые типы данных* поддерживают только целые числа (дробные части и десятичные точки не допускаются). Над этими типами разрешается выполнять арифметические операции и применять к ним агрегирующие функции (определение максимального, минимального, среднего и суммарного значения столбца реляционной таблицы).

INTEGER или *INT* – целое, для хранения которого отводится, как правило, 4 байта.

Примечание. Число байт, отводимое для хранения того или иного числового типа данных, зависит от используемой СУБД и аппаратной платформы. В данном случае приводятся наиболее «типичные» значения. Интервал значений от –2147483647 до +2147483648.

3. *Вещественные типы данных DECIMAL* (*p, n*), где *p* – общее количество десятичных цифр, *n* – количество цифр после десятичной запятой, описывают числа с дробной частью.

4. *Денежные типы данных* описывают, естественно, денежные величины. Если ваша система такого типа данных не поддерживает, то используйте *DECIMAL* (*p, n*).

Для типа данных *MONEY* (*p, n*) все аналогично типу *DECIMAL* (*p, n*). Данный тип вводится только потому, что некоторые СУБД предусматривают для него специальные методы форматирования.

5. Типы данных *Дата и время* используются для хранения даты, времени и их комбинаций. В большинстве СУБД можно определять интервал между двумя датами, а также уменьшать или увеличивать дату на определенное количество времени:

• *DATE* – тип данных для хранения даты.

• *TIME* – тип данных для хранения времени.

6. *Двоичные типы данных BINARY* позволяют хранить данные любого объема в двоичном коде (изображения, исполняемые файлы и т. д.). Определения этих типов наиболее сильно различаются от системы к системе, часто используется ключевое слово *BINARY*.

Для всех типов данных имеется общее значение *NULL* – «не определено». Это значение имеет каждый элемент столбца до тех пор, пока в него не будут введены данные. При создании таблицы можно явно указать СУБД, могут ли элементы того или иного столбца иметь значения *NULL*, так как это не допустимо, например, для столбца, являющегося первичным ключом.

5.2. DDL – язык определения данных

При описании команд языка DDL предполагается, что:

• текст, набранный прописными буквами (например, *CREATE TABLE*) используется для записи зарезервированных слов языка;

• текст, набранный строчными буквами и заключенный в угловые скобки (<*имя_базы_данных*>), обозначает переменную, вводимую пользователем;

• в квадратные скобки, например, [*NOT NULL*], заключается необязательная часть команды;

• взаимоисключающие элементы команды разделяются вертикальной чертой, например, [*UNIQUE* | *PRIMARY KEY*];

• многоточие (…) используется для указания необязательной возможности повторения конструкции, от нуля до нескольких раз.

5.2.1. Создание таблицы

Создание таблицы выполняется с помощью следующей команды языка *SQL*:

```
CREATE TABLE <имя_таблицы>
   (<имя_столбца> <тип_столбца>
      [NOT NULL]
      [UNIQUE | PRIMARY KEY]
      [REFERENCES <имя_мастер_таблицы> [<имя_столбца>]], ...
   );
```

При использовании таблицы пользователь обязан указать имя таблицы и список столбцов. Для каждого столбца обязательно указываются его имя и тип, а также могут быть указаны следующие параметры:

- *NOT NULL*. В этом случае элементы столбца всегда должны иметь определенное значение, отличное от *NULL*.
- *UNIQUE* – один из взаимоисключающих параметров. Значение каждого элемента столбца должно быть уникальным или *PRIMARY KEY* – столбец, являющийся первичным ключом.
- *REFERNECES <имя_мастер_таблицы>* [*<имя_столбца>*]. Эта конструкция определяет, что данный столбец является внешним ключом и ссылается на столбец первичного ключа *мастер_таблицы* (главной таблицы).

Контроль за выполнением указанных условий осуществляет СУБД.

Пример: создание базы данных УчетЗаказов и ее таблиц

```
CREATE DATABASE УчетЗаказов;
CREATE TABLE Клиент(КодКлиента VARCHAR(50) PRIMARY KEY,
   НаимКлиент VARCHAR(50) NOT NULL,
   КонтактЛицо VARCHAR(50) NOT NULL,
   Адрес VARCHAR(50) NOT NULL);
CREATE TABLE Товар(КодТов VARCHAR(50) PRIMARY KEY,
   НаимТовар VARCHAR(50) NOT NULL,
   ЕдИзм VARCHAR(50) NOT NULL,
   Цена INT,
   ИсхКолич INT);
CREATE TABLE Заказ(НомерЗаказа VARCHAR(50) PRIMARY KEY,
   ДатаЗаказа DATE NOT NULL,
   КодКлиента VARCHAR(50) REFERENCES Клиент(КодКлиента));
CREATE TABLE СтрокаЗаказа(НомерЗаказа VARCHAR(50) REFERENCES Заказ (НомерЗаказа),
   КодКлиента VARCHAR(50) REFERENCES Клиент (КодКлиента),
   Количество INT);
```

5.2.2. Удаление таблицы

Удаление таблицы выполняется с помощью команды

```
DROP TABLE <имя_таблицы>.
```

5.2.3. Модификация структуры таблицы

Для того, чтобы добавить столбцы, нужно выполнить следующую команду:

```
ALTER TABLE <имя_таблицы> ADD (<имя_столбца> <тип_столбца> [NOT NULL] [UNIQUE |
PRIMARY KEY] [REFERENCES <имя_мастер_таблицы> [<имя_столбца>]], ... );
```

Для удаления столбцов используется команда:

```
ALTER TABLE <имя_таблицы> DROP (<имя_столбца>,...);
```

Для модификации типа столбцов выполните следующую команду:

```
ALTER TABLE <имя_таблицы> MODIFY (<имя_столбца> <тип_столбца> [NOT NULL] [UNIQUE |
PRIMARY KEY] [REFERENCES <имя_мастер_таблицы> [<имя_столбца>]], ... );
```

5.3. DML – язык манипулирования данными

Язык DML является полнофункциональным языком манипулирования данными, который может использоваться как для выборки данных из базы, так и для модификации ее содержимого.

В *DML* имеются следующие операторы:

- *SELECT* – выборка данных из базы.
- *INSERT* – вставка данных в таблицу.
- *UPDATE* – обновление (изменение) данных в таблице.
- *DELETE* – удаление данных из таблицы.

5.3.1. Добавление новой записи в таблицу

Операторы модификации данных *INSERT, UPDATE* и *DELETE* наиболее простые, поэтому рассмотрим их в первую очередь.

Так, оператор данных *INSERT* выглядит следующим образом:

INSERT INTO <имя_таблицы> [(<имя_столбца>,
<имя_столбца>,...)]VALUES (<значение>, <значение>,..);

Список столбцов в данной команде не является обязательным параметром. В этом случае должны быть указаны значения для всех полей таблицы в том порядке, в котором эти столбцы были перечислены в команде CREATE TABLE, например:

INSERT INTO Товар VALUES ("ЯС","Яблоки сушеные", "кг", 5000, 100);

Примеры:

INSERT INTO Товар (КодТов, НаимТовар, ЕдИзм) VALUES ("ЯС","Яблоки сушеные", "кг");

INSERT INTO Товар (КодТов, НаимТовар, ЕдИзм) VALUES ("К5","Коньяк*****", "бут");

Примечание. Последний оператор необходимо реализовать в среде СУБД, так как его результат будет использован в примере 10 при рассмотрении оператора *SELECT*. На рис. 57 приведен результат преобразования в среде СУБД Access данного *SQL*-запроса в запрос на добавление в режим *Конструктор*.

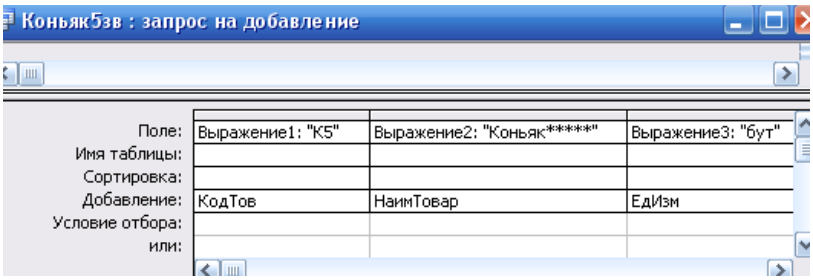


Рис. 57. **Запрос на добавление новой записи в таблицу *Товар***

Существует еще вторая форма оператора *INSERT*, которая позволяет скопировать множество строк одной таблицы в другую таблицу:

INSERT INTO <имя_таблицы> [(<имя_столбца>,
<имя_столбца>,...)]SELECT...;

5.3.2. Модификация записей

Для модификации записей используют оператор данных UPDATE:

UPDATE <имя_таблицы> SET <имя_столбца>=<значение>,...
[WHERE <условие>];

Если задано ключевое слово *WHERE* и условие, то команда *UPDATE* применяется только к тем записям, для которых оно выполняется. Если условие не задано, *UPDATE* применяется ко всем записям. Пример:

UPDATE Товар SET Цена=5500 WHERE КодТов = "ЯС";

В качестве условия используются логические выражения над константами и полями. В условиях допускаются следующие операции:

- операции сравнения: >, <, >=, <=, =, <>; в языке *SQL* эти операции могут применяться не только к числовым значениям, но и к строкам, так «<» означает раньше, а «>» – позже в алфавитном (лексикографическом) порядке и датам («<» – раньше и «>» – позже в хронологическом порядке);
- операции проверки поля на значение *NULL – IS NULL, IS NOT NULL*;
- операции проверки на вхождение в диапазон – *BETWEEN* и *NOT BETWEEN*;
- операции проверки на вхождение в список – *IN* и *NOT IN*;
- операции проверки на вхождение подстроки – *LIKE* и *NOT LIKE*;
- отдельные операции соединяются связями *AND, OR, NOT* и группируются с помощью круглых скобок.

Эти ключевые слова будут проиллюстрированы ниже в пункте 5.3.4, посвященном оператору *SELECT*.

Пример 9. UPDATE Товар SET Цена=4000 WHERE Цена IS NULL;

Эта команда находит в таблице *Товар* все неопределенные значения столбца *Цена* и заменяет их на 4000.

5.3.3. Удаление записей

DELETE FROM <имя_таблицы> [WHERE <условие>];

Удаляются все записи, удовлетворяющие указанному условию. Если ключевое слово WHERE и условие отсутствуют, из таблицы удаляются все записи. Пример:

DELETE FROM Товар WHERE Цена > 8000;

Эта команда удаляет записи о товарах с ценой дороже 8000.

5.3.4. Выборка данных из базы

Для извлечения записей из таблиц в *SQL* определен оператор *SELECT*. С помощью этой команды осуществляется не только операция реляционной алгебры «выборка» (горизонтальное подмножество), но и предварительное соединение (*join*) двух и более таблиц. Это наиболее сложное и мощное средство *SQL*. Полный синтаксис оператора *SELECT* имеет следующий вид:

SELECT [ALL | DISTINCT] <список_выбора>
FROM <имя_таблицы>, ...
[WHERE <условие>]
[GROUP BY <имя_столбца>, ...]
[HAVING <условие>]
[ORDER BY <имя_столбца> [ASC | DESC], ...]

Данный оператор всегда начинается с ключевого слова *SELECT* и состоит из шести разделов (предложений): *SELECT, FROM, WHERE,* GROUP *BY, HAVING* и *ORDER BY*. Только два из них – *SELECT* и *FROM* – являются обязательными.

Если присутствуют все шесть разделов оператора, то они обрабатываются в следующем порядке:

1. *FROM*. Определяются имена используемых одной или нескольких таблиц.
2. *WHERE*. Выполняется прореживание строк таблицы, полученной в предыдущем разделе.
3. *GROUP BY*. Образуются группы строк, имеющих одно и то же значение в указанном столбце.
4. *HAVING*. Выполняется прореживание групп строк, полученных в предыдущем разделе.
5. *SELECT*. Устанавливается, какие столбцы должны присутствовать в выходных данных.
6. *ORDER BY*. Производится сортировка результатов выполнения оператора.

Порядок разделов в операторе *SELECT* должен строго соблюдаться (например, *GROUP BY* должен всегда предшествовать *ORDER BY* и в записи оператора, и при его обработке), иначе это приведет к появлению ошибок.

Мы начнем рассмотрение *SELECT* с наиболее простых его форм. Все примеры запросов, приведенные ниже, касаются базы данных *УчетЗаказов*.

Примечание. Номерам приведенных примеров соответствуют номера запросов в среде СУБД Access.

Пример 10. Получить список всех наименований товаров.

SELECT НаимТовар FROM Товар;

На рисунках 58 и 59 приведены, соответственно, результат преобразования данного *SQL*-запроса в первом запросе в режиме *Конструктор* и результат первого запроса в среде СУБД Access.

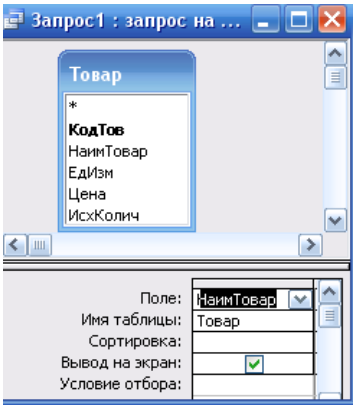


Рис. 58. **Первый запрос в режиме *Конструктор***

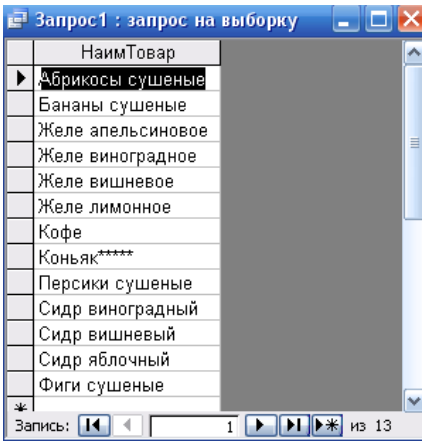


Рис. 59. **Результат первого запроса**

Пример 11. Получить список всех полей таблицы *Товар* (рис. 60):

SELECT * FROM Товар;

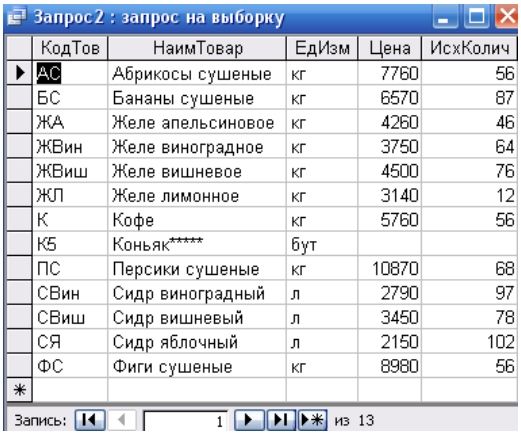


Рис. 60. **Результат второго запроса**

В том случае, когда нас интересуют не все записи, а только те, которые удовлетворяют некоторому условию, это условие можно указать после ключевого слова *WHERE*.

Пример 12. Найти все товары, цена которых больше 5000:

SELECT НаимТовар, Цена FROM Товар WHERE Цена > 5000;

На рисунках 61 и 62 представлены результаты преобразования данного *SQL*-запроса в режиме *Конструктор* и результат запроса.

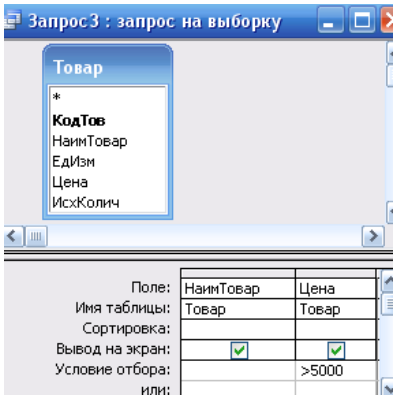


Рис. 61. **Третий запрос в режиме *Конструктор***

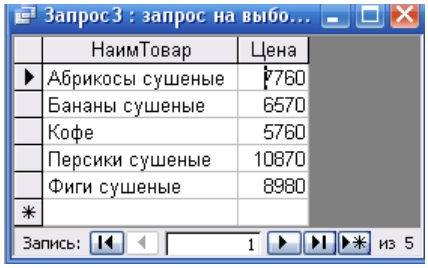


Рис. 62. **Результат третьего запроса**

Пример 13. Допустим теперь, что нам надо найти все товары в ценовом диапазоне от 4000 до 7000. Это условие можно записать в виде:

SELECT НаимТовар, Цена FROM Товар WHERE Цена >=4000 AND Цена <=7000;

На рисунках 63 и 64 приведены результаты преобразования данного запроса в режиме *Конструктор* и результат запроса.

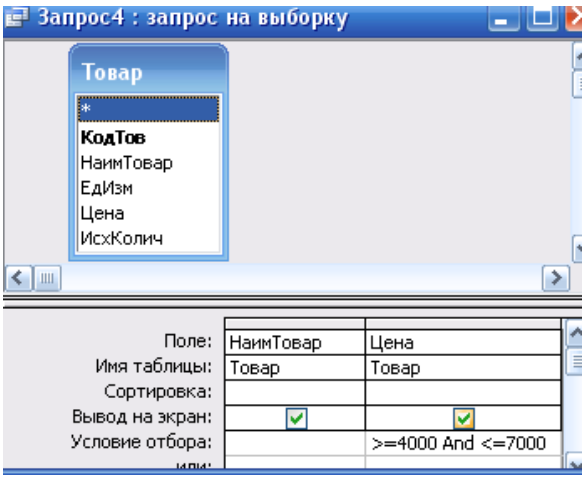


Рис. 63. **Четвертый запрос в режиме *Конструктор***

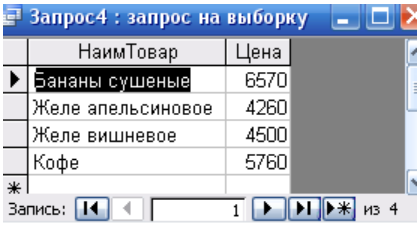


Рис. 64. **Результат четвертого запроса**

Пример 14. Другой вариант этой команды можно получить с использованием логической операции проверки на вхождение в интервал (рис. 65):

SELECT НаимТовар, Цена FROM Товар WHERE Цена BETWEEN 4000 AND 7000;

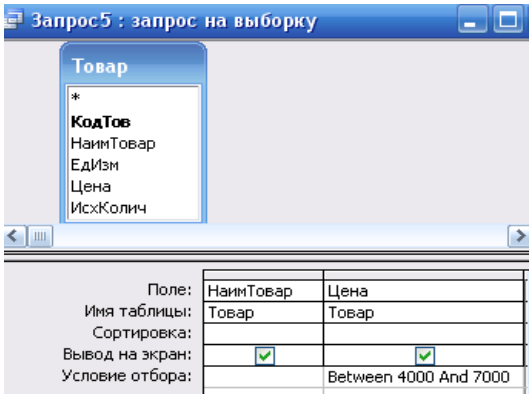


Рис. 65. **Пятый запрос в режиме *Конструктор***

При использовании конструкции *NOT BETWEEN* находятся все строки, не входящие в указанный диапазон.

Конструкция *IN* используется для проверки на вхождение в список.

Пример 15. Найти контактные лица клиентов «Нимфа» и «Пьер и К»:

SELECT КонтактЛицо FROM Клиент WHERE НаимКлиент IN ("Нимфа", "Пьер и К").

На рисунках 66 и 67 представлены результаты преобразования данного запроса в конструкторе и результат запроса.

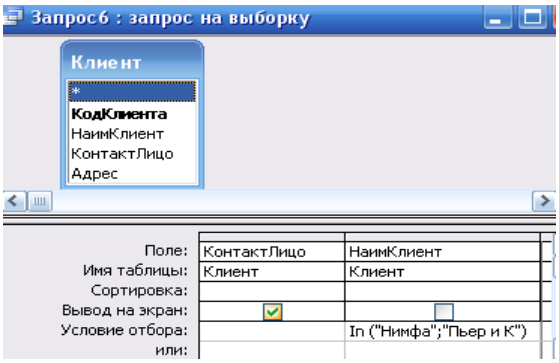


Рис. 66. **Шестой запрос в режиме *Конструктор***

Рис. 67. **Результат шестого запроса**

В данном примере в явном виде задан список интересующих нас значений. Конструкция *NOT IN* позволяет найти строки, не удовлетворяющие условиям, перечисленным в списке.

Наиболее полно преимущества ключевого слова *IN* проявляются во вложенных запросах, также называемых подзапросами. Предположим, нам нужно найти даты, когда выполнялись заказы фирмы «Нимфа». Наименования фирм содержатся в таблице *Клиент*, даты заказов в таблице *Заказ*. Ключевое слово *IN* позволяет объединить обе таблицы (без получения общего отношения) и извлечь при этом нужную информацию.

Пример 16. Найти даты, когда выполнялись заказы фирмы «Нимфа»:

SELECT ДатаЗаказа FROM Заказ WHERE КодКлиента *IN*
(SELECT КодКлиента FROM Клиент WHERE НаимКлиент= "Нимфа").

На рисунках 68 и 69 показаны результаты преобразования данного запроса в режиме *Конструктор* и результат запроса.

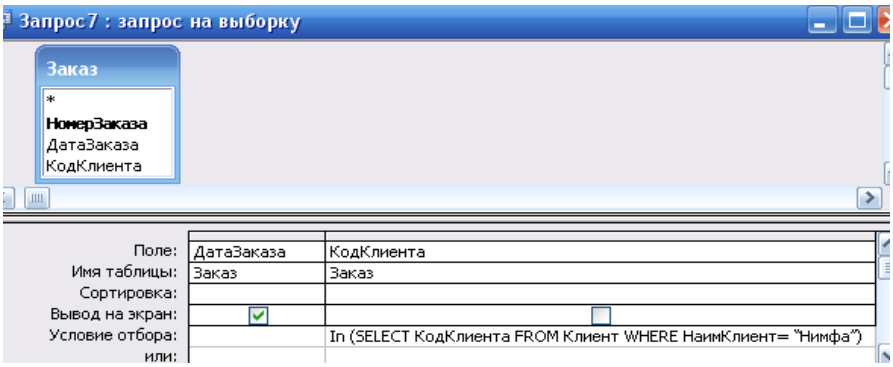


Рис. 68. **Седьмой запрос в режиме *Конструктор***

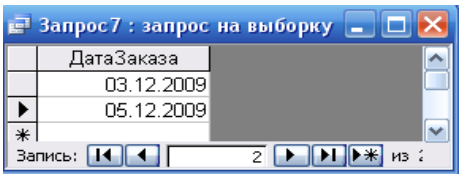


Рис. 69. **Результат седьмого запроса**

При выполнении этой команды СУБД вначале обрабатывает вложенный запрос по таблице *Клиент*, а затем его результат передает на вход основного запроса по таблице *Заказ*.

Некоторые задачи нельзя решить с использованием только операторов сравнения. Например, мы хотим найти адрес фирмы «Бубличная артель», но не знаем его точного наименования. Для решения этой задачи предназначено ключевое слово *LIKE*, его синтаксис в *SQL* имеет следующий вид:

WHERE <имя_столбца> LIKE <образец> [ESCAPE <ключевой_символ>].

Образец заключается в кавычки и должен содержать шаблон подстроки для поиска. Обычно в шаблонах используются два символа:

- знак процента (%) заменяет любое количество символов (в СУБД Access – это символ звездочка (*));
- подчеркивание (_) заменяет одиночный символ (в СУБД Access – символ вопросительный знак (?)).

Необязательная часть [*ESCAPE <ключевой_символ>*] используется, когда в образце для поиска имеются символы шаблона. В этом случае в среде СУБД MS Access, в отличие от стандарта *SQL*, вместо указанной необязательной части символы шаблона в образце просто заключаются в квадратные скобки (см. пример 10).

Пример 17. Найти адрес фирмы «Бубличная артель» по нескольким буквам в его наименовании, например, по фрагменту «арт»:

SELECT НаимКлиент, Адрес FROM Клиент WHERE НаимКлиент LIKE "*арт*".

На рисунках 70 и 71 представлены результаты преобразования данного *SQL*-запроса в режиме *Конструктор* и результат запроса в среде СУБД Access.

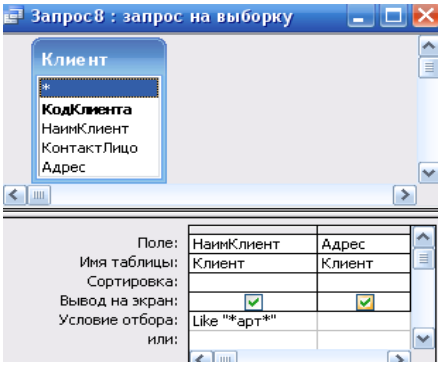


Рис. 70. **Восьмой запрос в режиме *Конструктор***

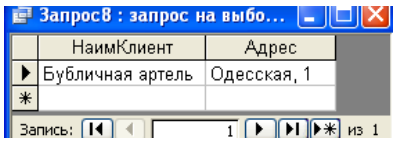


Рис. 71. **Результат восьмого запроса**

В соответствии с шаблоном СУБД найдет все строки, включающие в себя подстроку «арт».

Пример 18. Найти все товары, название которых начинается со слова "желе":

SELECT НаимТовар FROM Товар WHERE НаимТовар LIKE "желе*";

На рисунках 72 и 73 представлены результаты преобразования данного *SQL*-запроса в режиме *Конструктор* и результат запроса в среде СУБД Access.

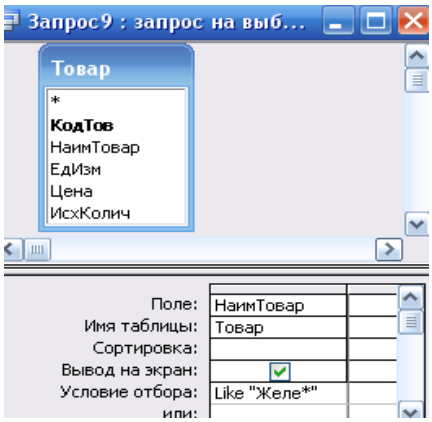


Рис. 72. **Девятый запрос в режиме *Конструктор***

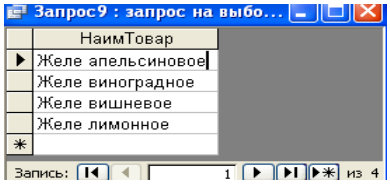


Рис. 73. **Результат девятого запроса**

Пример 19. Найти товары, в наименованиях которых есть символ «*», т. е. символ шаблона:

SELECT НаимТовар FROM Товар WHERE НаимТовар LIKE "*[*]*";

На рисунках 74 и 75 представлены результаты преобразования данных *SQL*-запроса в режиме *Конструктор* и результат запроса в среде СУБД Access.

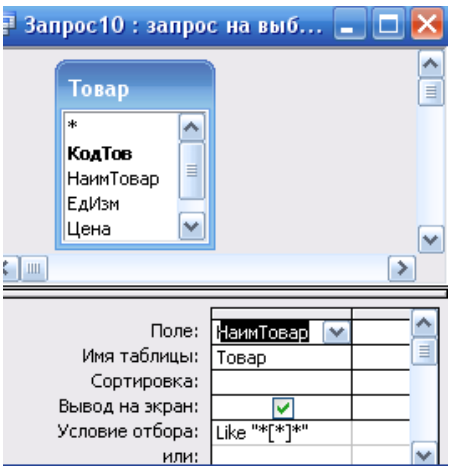


Рис. 74. **Десятый запрос в режиме *Конструктор***

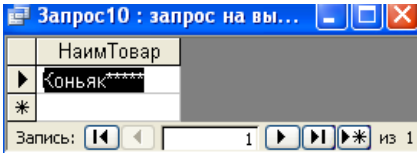


Рис. 75. **Результат десятого запроса**

В заключение заметим, что при выполнении оператора *SELECT* результирующее отношение может иметь несколько записей с одинаковыми значениями всех полей. Чтобы исключить повторяющиеся записи из выборки, используется ключевое слово *DISTINCT*. Ключевое слово *ALL* указывает, что в результат необходимо включать все строки.

5.3.5. Выборка из нескольких таблиц

Очень часто возникает ситуация, когда выборку данных надо производить из отношения, которое является результатом соединения (*join*) двух других отношений. Например, нам нужно получить из базы данных *УчетЗаказов* информацию обо всех заказах клиентов в виде таблицы, состоящей из трех столбцов (*НаимКлиент*, *НомерЗаказа*, *ДатаЗаказа*).

Для удобства на рис. 76 еще раз приведем схему данных БД *УчетЗаказов*.

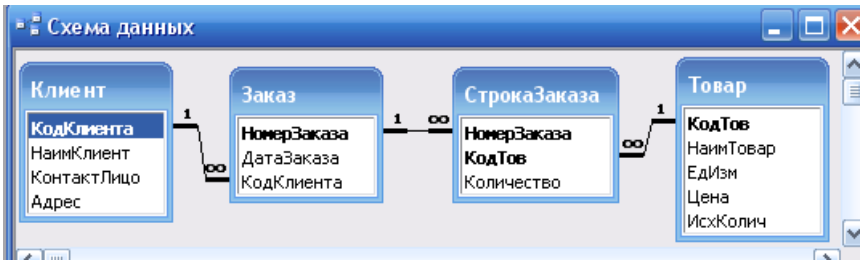


Рис. 76. **Схема данных БД *Учет заказов***

При решении задач подобного рода СУБД предварительно должна выполнить соединение таблиц, в данном случае таблиц *Клиент* и *Заказ*, а только затем произвести выборку из полученного отношения. Для этого в операторе *SELECT* после ключевого слова *FROM* указывается список таблиц, по которым производится поиск данных. После ключевого слова *WHERE* указывается условие, по которому производится соединение.

Пример 20. Найти данные обо всех заказах клиентов с указанием номера и даты каждого заказа:

```
SELECT Клиент.НаимКлиент, Заказ.НомерЗаказа, Заказ.ДатаЗаказа;
FROM Клиент, Заказ;
WHERE Клиент.КодКлиента = Заказ.КодКлиента.
```

На рисунках 77 и 78 представлены результаты преобразования данного *SQL*-запроса в режиме *Конструктор* и результат запроса в среде СУБД Access.

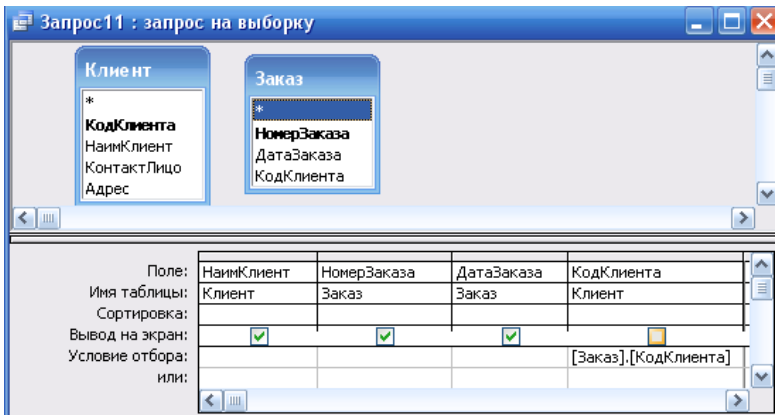


Рис. 77. **Одиннадцатый запрос в режиме *Конструктор***

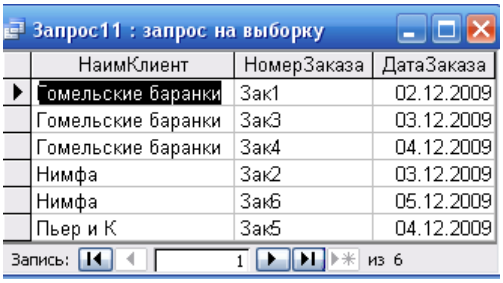


Рис. 78. **Результат одиннадцатого запроса**

Следует обратить внимание на то, что когда в разных таблицах присутствуют одноименные поля, то для устранения неоднозначности перед именем поля указывается имя таблицы и знак «.» (точка) *Имя таблицы следует указывать всегда*.

Пример 21. Результат предыдущего запроса ограничить заказами в интервале дат от 3 до 5 декабря 2009 г. включительно:

```
SELECT Клиент.НаимКлиент, Заказ.НомерЗаказа, Заказ.ДатаЗаказа
FROM Клиент, Заказ
WHERE Клиент.КодКлиента = Заказ.КодКлиента AND
    (Заказ.ДатаЗаказа BETWEEN #12/03/2009# AND #12/05/2009#);
```

На рисунках 79 и 80 представлены результаты преобразования данного *SQL*-запроса в режиме *Конструктор* и результат запроса в среде СУБД Access.

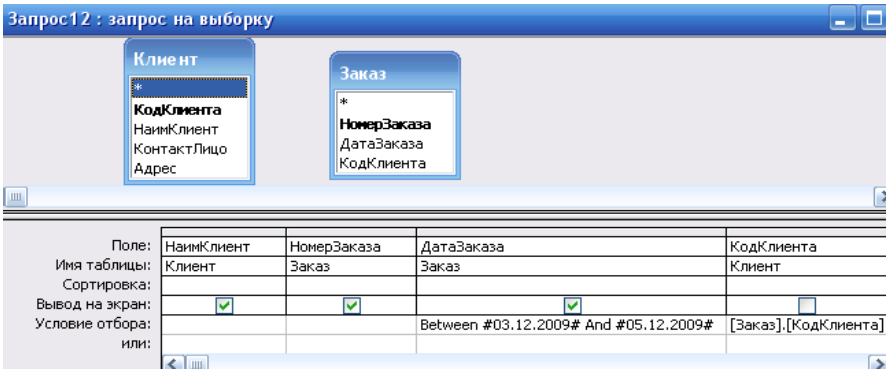


Рис. 79. **Двенадцатый запрос в режиме *Конструктор***

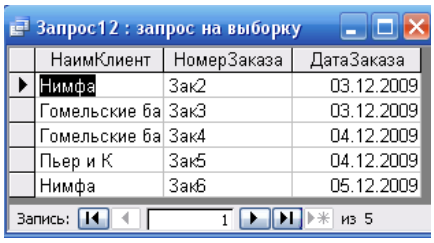


Рис. 80. **Результат двенадцатого запроса**

Следует отметить, что можно производить соединение и более чем двух таблиц.

Пример 22. Дополнить выборку предыдущего примера названиями заказанных товаров:

```
SELECT Клиент.НаимКлиент, Заказ.НомерЗаказа, Заказ.ДатаЗаказа, Товар.НаимТовар
FROM Клиент, Заказ, СтрокаЗаказа, Товар
WHERE Клиент.КодКлиента = Заказ.КодКлиента AND
    Заказ.НомерЗаказа = СтрокаЗаказа.НомерЗаказа AND
    СтрокаЗаказа.КодТов= Товар.КодТов AND
    (Заказ.ДатаЗаказа Between #12/3/2009# And #12/5/2009#);
```

На рисунках 81 и 82 представлены результаты преобразования данного *SQL*-запроса в режиме *Конструктор* и результат запроса в среде СУБД Access.

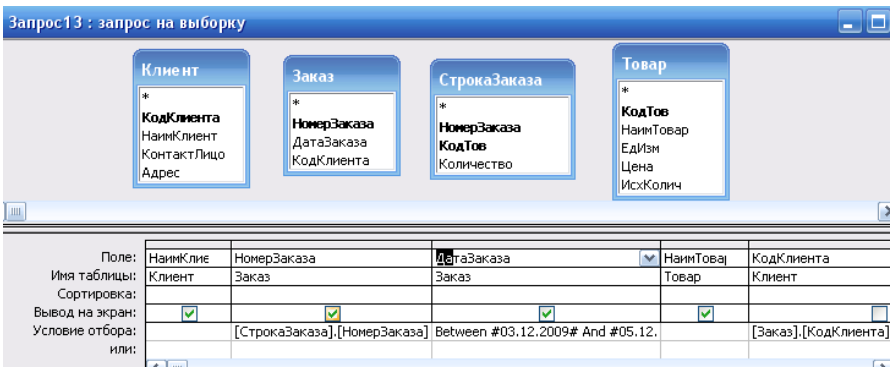


Рис. 81. **Тринадцатый запрос в режиме *Конструктор***

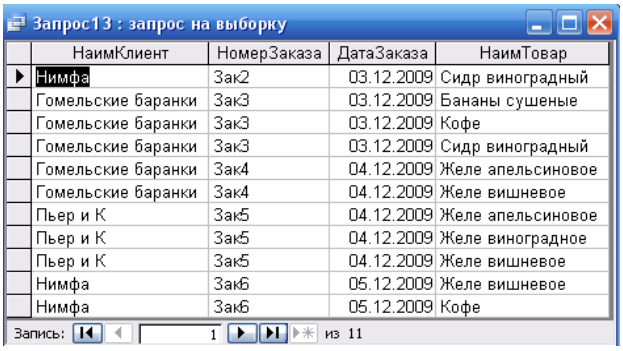


Рис. 82. **Результат тринадцатого запроса**

6. НОРМАЛИЗАЦИЯ

6.1. Универсальное отношение: избыточность и аномалии обновления

В подразделе 1.2 реляционная база данных была определена как совокупность отношений, содержащих всю информацию, которая должна храниться в БД. Может возникнуть естественный вопрос, а нельзя ли, чтобы база данных состояла лишь из одного отношения (из одной таблицы)? Действительно, в литературе приводятся примеры, когда некоторые крупные корпорации используют базы данных, состоящие из одного отношения, называемого универсальным отношением (или универсальной таблицей – *УТ*), в котором сосредоточены все представляющие интерес данные.

На рис. 83 приведена универсальная таблица, содержащая ту же информацию, т. е. те же атрибуты, что и база данных *Поставщики-Детали*:

УТПД (*Дкод*, *Дназв*, *Цвет*, *Вес*, *Шт*, *Пфам*, *Город*).

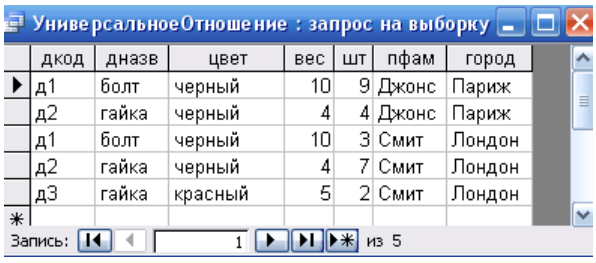


Рис. 83. **Универсальная таблица БД *Поставщики-Детали***

Данная таблица получена естественным соединением всех трех таблиц *Деталь*, *Поставки* и *Поставщик* (рис. 84).

Несмотря на внешнюю привлекательность представления базы данных в виде одного отношения, этот подход считается неприемлемым по следующим причинам:

1. Слишком большой объем избыточных данных, хранимых в универсальной таблице.
2. Присущие универсальной таблице аномалии обновления.

Избыточность данных проявляется в том, что вслед за кодом детали выписываются значения атрибутов детали во всех строках, где эта деталь встречается. Аналогично, вслед за фамилией поставщика записывается город его проживания во всех строках, где встречается этот поставщик.

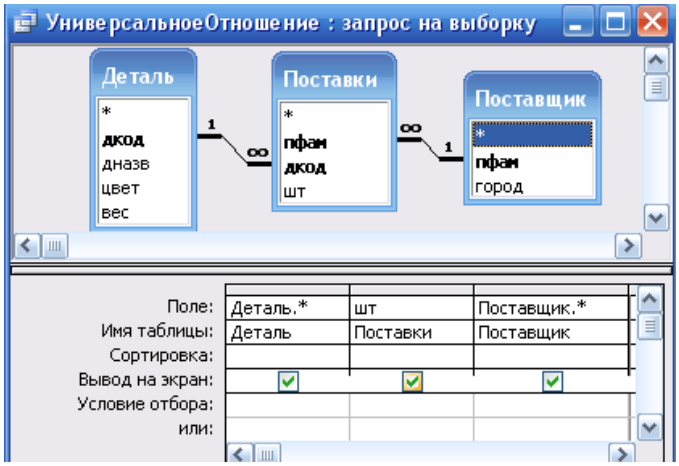


Рис. 84. **Создание таблицы, включающей все поля БД *Поставщики-Детали***

Аномалии обновления универсальной таблицы подразделяются на аномалии вставки, удаления и модификации.

Аномалии *вставки* проявляются, например, в том, что при вводе информации о новой детали необходимо также вводить информацию о поставщике этой детали. Если же эта деталь еще не поставлялась, то в столбцах *пфам* и *город* появятся пустые значения, которые нежелательны. Более того, если первичным ключом *УТПД* является пара полей <***дкод, пфам***>, то указанную запись вообще не удастся ввести из-за нарушения целостности сущности.

Аномалия *удаления* проявляется в том, что при удалении последней записи с информацией о единственной поставке какой-либо детали, сведения об этой детали будут полностью удалены из базы данных, хотя эти сведения могли бы потребоваться в дальнейшем. Например, при удалении последней строки из таблицы *УТПД*, из базы данных исчезнет полностью информация о детали *д*3.

Аномалии *модификации* проявляются в том, что изменения, касающиеся, например, какого-то поставщика, должны быть одновременно отражены во всех строках, где этот поставщик встречается. Например, при изменении города проживания у Смита соответствующие изменения должны быть одновременно внесены в три последние записи *УТПД*. Или при изменении цвета детали *д*1 с черного на желтый соответствующие изменения должны быть произведены одновременно в первой и третьей строках *УТПД*.

Для преодоления перечисленных недостатков универсальной таблицы ее подвергают декомпозиции, т. е. разбивают на отношения с меньшей степенью. Этот процесс последовательной декомпозиции отношений проектируемой базы данных, начиная с универсальной таблицы, называется нормализацией.

6.2. Нормальные формы и функциональные зависимости

В теории реляционных баз данных определена следующая последовательность нормальных форм отношений:

- первая нормальная форма (1НФ);
- вторая нормальная форма (2НФ);
- третья нормальная форма (3НФ);
- нормальная форма Бойса-Кодда (НФБК);
- четвертая нормальная форма (4НФ);
- пятая нормальная форма (5НФ).

Нормальные формы обладают следующими свойствами:

- каждая следующая нормальная форма отношений должна обладать «улучшенными» свойствами по сравнению с предыдущей нормальной формой (в том смысле, что у них будут устраняться присущие универсальной таблице недостатки);
- при переходе к следующей нормальной форме свойства предыдущих нормальных форм будут сохраняться.

Как видно из рис. 85, часть отношений в 1НФ находится в 2НФ, часть отношений в 2НФ находится в 3НФ и т. д.

Обычно считается, что отношения реляционной базы данных, в том числе универсальная таблица, всегда находятся, как минимум, в 1НФ. Аномалии обновления, требующие, чтобы отношения находились в 4НФ или 5НФ, встречаются крайне редко. Поэтому на практике процесс нормализации отношений БД стараются довести до НФБК или, по крайней мере, до 3НФ. Это вызвано также тем обстоятельством, что нормальные формы отношений 2НФ, 3НФ и НФБК основываются на одном понятии функциональных зависимостей (ФЗ).

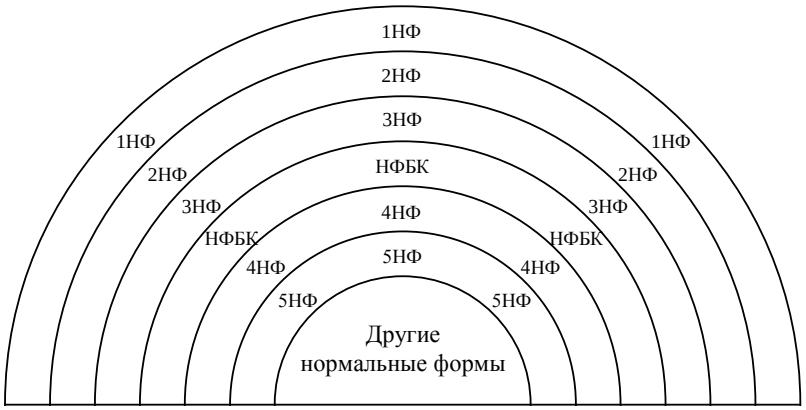


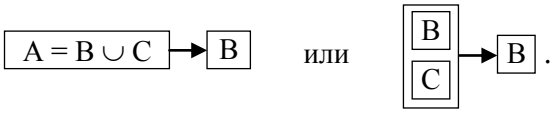
Рис. 85. **Схема взаимосвязей нормальных форм отношений**

Говорят, что атрибут B *функционально зависит* от атрибута A, если для каждого значения A существует ровно одно связанное с ним значение B (в любой момент времени). A и B могут быть составными, т. е. они могут представлять собой не единичные атрибуты, а группы, состоящие из двух и более атрибутов. Обозначение $A \to B$. Графическое обозначение [A] → [B].

Множество всех функциональных зависимостей, которые могут быть выведены из заданного множества функциональных зависимостей X, называется замыканием X и записывается как X^+. Существуют следующие три правила вывода, называемые аксиомами Армстронга и позволяющие вычислить X^+ из X:

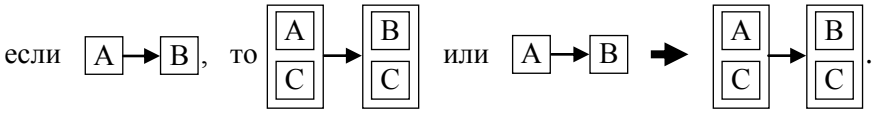
1. *Рефлексивность*. Если B – подмножество A, то $A \to B$.

Графическое обозначение



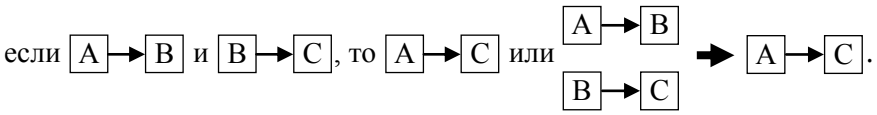
2. *Дополнение*. Если $A \to B$, то $(A, C) \to (B, C)$.

Графическое обозначение



3. *Транзитивность*. Если $A \to B$ и $B \to C$, то $A \to C$.

Графическое обозначение

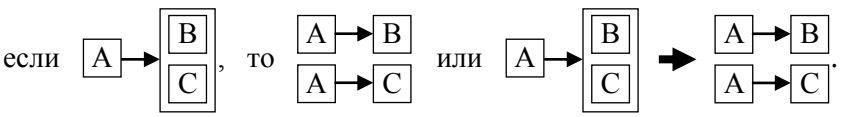


Из аксиом Армстронга могут быть выведены следующие полезные правила:

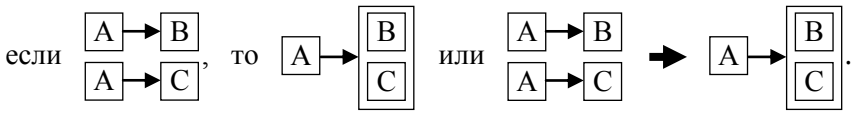
1. *Самоопределение*. $A \to A$ или [A] → [A].

2. *Декомпозиция*. Если $A \to (B, C)$, то $A \to B$ и $A \to C$.

Графическое обозначение



3. *Объединение*. Если A → B и A → C, то A → (B, C).
Графическое обозначение



4. *Композиция*. Если A → B и C → D, то (A, C) → (B, D).
Графическое обозначение

Для универсальной таблицы БД *Поставщики-Детали* УТПД имеют место следующие ФЗ:
ФЗ 1.1: *Дкод* → (*Дназв*, *Цвет*, *Вес*)

ФЗ 1.2: *Пфам* → *Город*

ФЗ 1.3: (***Дкод***, ***Пфам***) → *Шт*
ФЗ 1.1 получена как результат применения правила 6 – объединения, к трем следующим ФЗ:

Дкод → *Дназв*,

Дкод → *Цвет*,

Дкод → *Вес*.

Графически эти ФЗ представлены на рисунках 86 и 87.

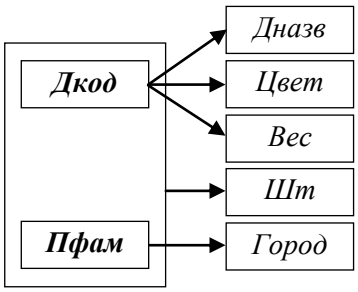


Рис. 86. **Исходные ФЗ для УТПД**

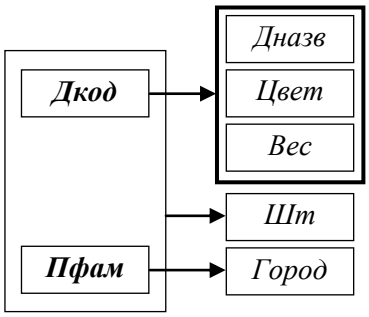


Рис. 87. **ФЗ для УТПД после применения правила 6 (объединение)**

На рисунках 88 и 89 представлена универсальная таблица БД *УчетЗаказов* УТУЗ:

УТУЗ (***КодТов***, *НаимТов*, *ЕдИзм*, *Цена*, *ИсхКолич*, *КодКлиента*, *НаимКлиент*, *КонтактЛицо*, *Адрес*, ***НомерЗаказа***, *ДатаЗаказа*, *Количество*).
Для УТУЗ имеют место следующие ФЗ:

ФЗ 2.1: *КодТов* → (*НаимТов*, *ЕдИзм*, *Цена*, *ИсхКолич*)

ФЗ 2.2: *КодКлиента* → (*НаимКлиент*, *КонтактЛицо*, *Адрес*)

ФЗ 2.3: *НомерЗаказа* → (*ДатаЗаказа*, *КодКлиента*)

ФЗ 2.4: (***КодТов***, ***НомерЗаказа***) → *Количество*

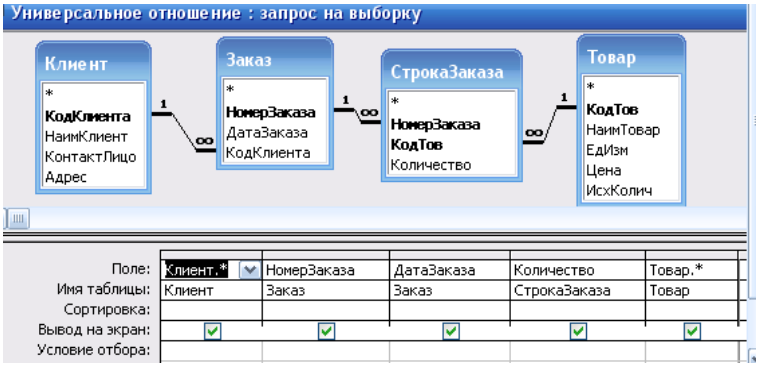
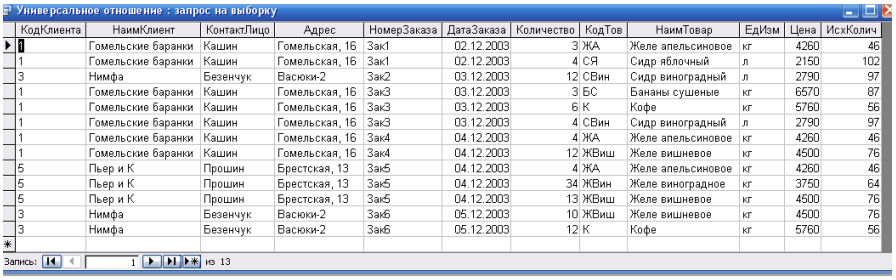


Рис. 88. **Создание таблицы, включающей все поля БД *УчетЗаказов***


Универсальное отношение : запрос на выборку

КодКлиента	НаимКлиент	КонтактЛицо	Адрес	НомерЗаказа	ДатаЗаказа	Количество	КодТов	НаимТовар	ЕдИзм	Цена	ИсхКолич
1	Гомельские баранки	Кашин	Гомельская, 16	Зак1	02.12.2003	3	ЖА	Желе апельсиновое	кг	4260	46
1	Гомельские баранки	Кашин	Гомельская, 16	Зак1	02.12.2003	4	СЯ	Сидр яблочный	л	2150	102
3	Нимфа	Безенчук	Васюки-2	Зак2	03.12.2003	12	СВин	Сидр виноградный	л	2790	97
1	Гомельские баранки	Кашин	Гомельская, 16	Зак3	03.12.2003	3	БС	Бананы сушеные	кг	6570	87
1	Гомельские баранки	Кашин	Гомельская, 16	Зак3	03.12.2003	6	К	Кофе	кг	5760	56
1	Гомельские баранки	Кашин	Гомельская, 16	Зак3	03.12.2003	4	СВин	Сидр виноградный	л	2790	97
1	Гомельские баранки	Кашин	Гомельская, 16	Зак4	04.12.2003	4	ЖА	Желе апельсиновое	кг	4260	46
1	Гомельские баранки	Кашин	Гомельская, 16	Зак4	04.12.2003	12	ЖВиш	Желе вишневое	кг	4500	76
5	Пьер и К	Прошин	Брестская, 13	Зак5	04.12.2003	4	ЖА	Желе апельсиновое	кг	4260	46
5	Пьер и К	Прошин	Брестская, 13	Зак5	04.12.2003	34	ЖВин	Желе виноградное	кг	3750	64
5	Пьер и К	Прошин	Брестская, 13	Зак5	04.12.2003	13	ЖВиш	Желе вишневое	кг	4500	76
3	Нимфа	Безенчук	Васюки-2	Зак6	05.12.2003	10	ЖВиш	Желе вишневое	кг	4500	76
3	Нимфа	Безенчук	Васюки-2	Зак6	05.12.2003	12	К	Кофе	кг	5760	56

Запись: 1 из 13

Рис. 89. **Универсальная таблица БД *УчетЗаказов* (УТУЗ)**

Определение 2НФ основано на понятии *полной функциональной зависимости*: атрибут *B* находится в *полной функциональной зависимости* от атрибута *A*, если атрибут *B* является функционально зависимым от *A*, но не зависит ни от одного собственного подмножества атрибута *A*.

Функциональная зависимость A → B является полной функциональной зависимостью, если удаление какого-либо атрибута из A приводит к утрате этой зависимости.

Функциональная зависимость A → B называется частичной, если в *A* есть некий атрибут, при удалении которого эта зависимость сохраняется.

Вторая нормальная форма (2НФ).

Отношение, которое находится в первой нормальной форме и каждый атрибут которого, не входящий в состав первичного ключа, характеризуется полной функциональной зависимостью от этого первичного ключа.

Для проверки принадлежности отношения к 2НФ необходимо убедиться в отсутствии в нем частичных ФЗ от первичного ключа.

Таблица *УТПД* не находится в 2НФ, так как ФЗ 1.1 и ФЗ 1.2 не удовлетворяют условию полной функциональной зависимости (от первичного ключа <***дкод***, ***пфам***>). То же можно сказать и о таблице УТУЗ.

Определение 3НФ основано на понятии транзитивной функциональной зависимости. Например, в УТУЗ имеется следующая транзитивная ФЗ:

НомерЗаказа → *КодКлиента* → (*НаимКлиент*, *КонтактЛицо*, *Адрес*).

Третья нормальная форма (3НФ).

Отношение, которое находится во второй нормальной форме и не имеет атрибутов, не входящих в первичный ключ, которые находились бы в транзитивной функциональной зависимости от этого первичного ключа.

Для проверки принадлежности (уже находящегося в 2НФ) отношения к 3НФ необходимо убедиться в отсутствии в нем транзитивных ФЗ от первичного ключа.

Определение нормальной формы Бойса-Кодда основано на понятии детерминанта функциональной зависимости: детерминантом функциональной зависимости называется атрибут или группа атрибутов, расположенная слева от стрелки.

Нормальная форма Бойса-Кодда (НФБК).

Отношение находится в НФБК тогда и только тогда, когда каждый его детерминант является потенциальным ключом.

Для проверки принадлежности отношения к НФБК необходимо найти все его детерминанты и убедиться в том, что они являются потенциальными ключами.

НФБК является более строгой формой по сравнению с 3НФ, т. е. каждое отношение в НФБК является также отношением в 3НФ, но не всякое отношение в 3НФ является отношением в НФБК. Однако различия между этими формами проявляются на практике крайне редко и только при выполнении следующих условий:

- имеются два (или несколько) составных потенциальных ключа;
- эти потенциальные ключи перекрываются, т. е. ими совместно используется, по крайней мере, один общий атрибут.

6.3. Нормализация универсального отношения с использованием функциональных зависимостей

Один подходящий алгоритм нормализации отношений реляционной базы данных вплоть до НФБК состоит из следующих шагов:

1. В отношении R (A, B, C, D, E, …) выбирается ФЗ, например, C → D, которая является причиной того, что отношение R не находится в НФБК, т. е. атрибут или группа атрибутов C не является потенциальным ключом отношения R. Если таких ФЗ нет, то R находится в НФБК и его нормализация завершена.

2. Для ФЗ C → D выделяется отдельное отношение R1 (C, D) с первичным ключом C, а из отношения R удаляется атрибут (или группа атрибутов) D. Таким образом, отношение R (A, B, C, D, E, …) разбивается (декомпозируется) на два отношения R1 (C, D) и R2 (A, B, C, E, …).

3. Первый и второй шаги повторяются теперь уже для отношения R2, и т. д.

Если в декомпозируемом отношении R имеются цепочки ФЗ типа C → D → E, то на шаге 1 для отдельного отношения необходимо выбирать ФЗ из конца цепочки, т. е. ФЗ D → E.

Покажем реализацию этого алгоритма вначале на примере *УТПД*, затем – на примере *УТУЗ*.

6.3.1. Нормализация универсальной таблицы УТПД

В отношении УТПД (***Дкод***, *Дназв*, *Цвет*, *Вес*, *Шт*, ***Пфам***, *Город*) детерминант ФЗ 1.1, т. е. атрибут *Дкод*, не является потенциальным ключом. Следовательно, в соответствии с вышеизложенным алгоритмом, эта ФЗ может служить основой для декомпозиции УТПД на два отношения:

- R1 (***Дкод***, *Дназв*, *Цвет*, *Вес*);
- R2 (***Дкод***, *Шт*, ***Пфам***, *Город*).

ФЗ для отношений *R*1 и *R*2 представлены на рис. 90.

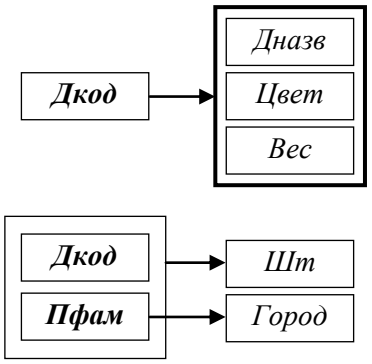


Рис. 90. **Функциональные зависимости для отношений R1 и R2**

В отношении R2 (***Дкод***, *Шт*, ***Пфам***, *Город*) детерминант ФЗ 1.2, т. е. атрибут Пфам, не является потенциальным ключом. Следовательно, эта ФЗ может служить основой для декомпозиции R2 на два отношения:

- R21 (***Пфам***, *Город*);
- R22 (***Дкод***, *Шт*, ***Пфам***).

В итоге исходное отношение УТПД (***Дкод***, *Дназв*, *Цвет*, *Вес*, *Шт*, ***Пфам***, *Город*) оказалось разбитым на три отношения:

- R1 (***Дкод***, *Дназв*, *Цвет*, *Вес*);
- R21 (***Пфам***, *Город*);
- R22 (***Дкод***, *Шт*, ***Пфам***).

Каждое из отношений R1, R21, R22 находится в НФБК, так как детерминантами всех ФЗ в них являются потенциальные ключи (в каждом из этих отношений один потенциальный ключ, совпадающий с пер-

вичным ключом). Переименовав названия этих отношений, получим окончательный состав таблиц базы данных *Поставщики-Детали* и состав атрибутов таблиц:

- Деталь (***Дкод***, *Дназв*, *Цвет*, *Вес*);
- Поставщик (***Пфам***, *Город*);
- Поставки (***Дкод***, *Шт*, ***Пфам***).

ФЗ для отношений *Деталь*, *Поставщик* и *Поставки* представлены на рис. 91.

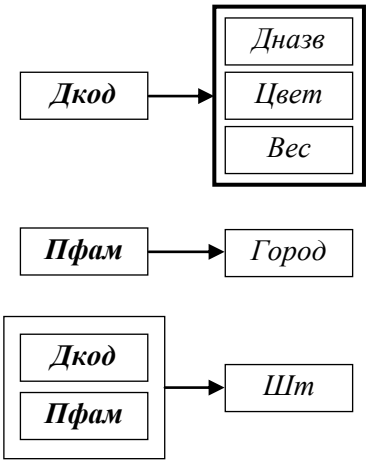


Рис. 91. **ФЗ для отношений *Деталь*, *Поставщик* и *Поставки***

Можно убедиться, что перечисленные в подразделе 6.1 чрезмерная избыточность и аномалии обновления в полученной базе данных устранены.

6.3.2. Нормализация универсальной таблицы УТУЗ

В отношении

УТУЗ (*КодТов*, *НаимТов*, *ЕдИзм*, *Цена*, *ИсхКолич*, *КодКлиента*, *НаимКлиент*, *КонтактЛицо*, *Адрес*, *НомерЗаказа*, *ДатаЗаказа*, *Количество*)

из ФЗ 2.3 НомерЗаказа → (ДатаЗаказа, КодКлиента) по пятому правилу «декомпозиция» следует справедливость ФЗ:

НомерЗаказа → *КодКлиента*.

Следовательно, имеется цепочка транзитивной функциональной зависимости:
НомерЗаказа → *КодКлиента* → (*НаимКлиент*, *КонтактЛицо*, *Адрес*).
В соответствии с вышеизложенным алгоритмом, для декомпозиции УТУЗ удобно выбрать ФЗ из конца цепочки: *КодКлиента* → (*НаимКлиент*, *КонтактЛицо*, *Адрес*).
В результате получатся отношения:

- R1 (***КодКлиента***, *НаимКлиент*, *КонтактЛицо*, *Адрес*);
- R2 (***КодТов***, *НаимТов*, *ЕдИзм*, *Цена*, *ИсхКолич*, *КодКлиента*, ***НомерЗаказа***, *ДатаЗаказа*, *Количество*).

В отношении R2 детерминант ФЗ 2.1: *КодТов* → (*НаимТов*, *ЕдИзм*, *Цена*, *ИсхКолич*), т. е. атрибут *КодТов*, не является потенциальным ключом. Следовательно, эта ФЗ может служить основой для декомпозиции R2 на два отношения:

- R21 (***КодТов***, *НаимТов*, *ЕдИзм*, *Цена*, *ИсхКолич*);
- R22 (***КодТов***, *КодКлиента*, ***НомерЗаказа***, *ДатаЗаказа*, *Количество*).

В отношении R22 детерминант ФЗ 2.3: *НомерЗаказа* → (*ДатаЗаказа*, *КодКлиента*), т. е. атрибут *НомерЗаказа*, не является потенциальным ключом. Следовательно, эта ФЗ может служить основой для декомпозиции R22 на два отношения:

- R221 (***НомерЗаказа***, *ДатаЗаказа*, *КодКлиента*);
- R222 (***КодТов***, ***НомерЗаказа***, *Количество*).

В итоге исходное отношение УТУЗ (***КодТов***, *НаимТов*, *ЕдИзм*, *Цена*, *ИсхКолич*, *КодКлиента*, *НаимКлиент*, *КонтактЛицо*, *Адрес*, ***НомерЗаказа***, *ДатаЗаказа*, *Количество*) оказалось разбитым на четыре отношения:

- R1 (***КодКлиента***, *НаимКлиент*, *КонтактЛицо*, *Адрес*);

- R21 (***КодТов***, *НаимТов*, *ЕдИзм*, *Цена*, *ИсхКолич*);
- R221 (***НомерЗаказа***, *ДатаЗаказа*, *КодКлиента*);
- R222 (***КодТов***, ***НомерЗаказа***, *Количество*).

Каждое из отношений R1, R21, R221 и R222 находится в НФБК, так как детерминантами всех ФЗ в них являются потенциальные ключи (в каждом из этих отношений один потенциальный ключ, совпадающий с первичным ключом). Переименовав названия этих отношений, получим окончательный состав таблиц базы данных *УчетЗаказов* и состав атрибутов таблиц:

- Клиент (***КодКлиента***, *НаимКлиент*, *КонтактЛицо*, *Адрес*);
- Товар (***КодТов***, *НаимТов*, *ЕдИзм*, *Цена*, *ИсхКолич*);
- Заказ (***НомерЗаказа***, *ДатаЗаказа*, *КодКлиента*);
- СтрокаЗаказа (***КодТов***, ***НомерЗаказа***, *Количество*).

Порядок нормализации УТУЗ также можно проиллюстрировать графически. Сделать это предлагается студентам самостоятельно.

ЗАКЛЮЧЕНИЕ

В предисловии к восьмому изданию своей книги [1] К. Дейт выражал обеспокоенность таким большим объемом, который приобрела эта книга – 1328 страниц. Вместе с тем он отмечает, что технология баз данных стала настолько обширной областью знаний, что невозможно рассмотреть всю относящуюся к ней проблематику в книге объемом меньше 1000 страниц. В подтверждение этих слов можно отметить, что третье русскоязычное издание книги [2] приобрело объем в 1440 страниц вместо прежних 1120 страниц.

В данном пособии рассмотрены только самые начальные технологии реляционных баз данных, которые должны дать студентам представление об этой важнейшей области программного обеспечения. Только перечисление не упомянутых вопросов из теории баз данных могло бы занять не одну страницу.

Тем не менее, необходимо указать на особенности применения изложенных здесь подходов к проектированию баз данных методом «сущность–связь» и методом декомпозиции начального универсального отношения. Считается, что метод декомпозиции становится слишком громоздким, когда число атрибутов превышает 20 [3]. В этом случае вначале используется метод «сущность–связь», и только на конечном этапе проектирования полученные отношения базы данных проверяются на «нормальность». Как было сказано выше, проверяется выполнение условий НФБК или, в крайнем случае, выполнение условий 3НФ. Следует подчеркнуть, что устранение излишней избыточности и аномалий обновления отношений БД особенно актуальны для тех реляционных баз данных, которые используются в информационных системах оперативной обработки транзакций, так называемых OLTP-системах (On-Line Transaction Processing – OLTP). Примерами таких систем являются системы обслуживания клиентов банков, резервирования билетов на транспорте и мест в гостиницах и т. д.

Для систем поддержки принятия решений, так называемых DSS-систем (Decision Support System – DSS), указанные проблемы обновления не столь актуальны, так как информация в их базах данных не обновляется или обновляется очень редко. Обычно такие базы данных реализуются в виде хранилищ данных (Data Warehouse) и в них используются инструменты оперативной аналитической обработки (On-Line Analytical Processing – OLAP) и инструменты разработки данных (Data Mining). Однако и в этом случае следует иметь в виду, что оперативные данные для хранилищ данных поставляются из реляционных баз данных OLTP-систем. Поэтому и здесь четкое и ясное нормализованное представление исходных оперативных данных весьма важно.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. **Дейт, К. Дж.** Введение в системы баз данных / К. Дж. Дейт. – 8-е изд. : [пер. с англ.]. – М. : Вильямс, 2005. – 1328 с.

2. **Коннолли, Т.** Базы данных: проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг, А. Страчан : 2-е изд. : [пер. с англ.]. – М. : Вильямс, 2000. – 1120 с.

3. **Джексон, Г.** Проектирование реляционных баз данных для использования с микроЭВМ : [пер. с англ.] / Г. Джексон. – М. : Мир, 1991. – 252 с.

4. **ER-метод** проектирования баз данных и его реализация в среде СУБД Access : пособие для студентов экон. специальностей / авт.-сост. : С. М. Мовшович, К. Г. Сулейманов. – Гомель : Бел. торгово-экон. ун-т потребит. кооп., 2003. – 140 с.

СОДЕРЖАНИЕ

Учебное издание

СИСТЕМЫ БАЗ ДАННЫХ

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ И РАБОТА С НИМИ В СРЕДЕ СУБД MS ACCESS

Пособие
для студентов специальности 1-26 03 01
«Управление информационными ресурсами»

Авторы-составители:
Мовшович Семен Михайлович
Сулейманов Казбик Гамирович

Редактор О. А. Ивановская
Технический редактор Н. Н. Короедова
Компьютерная верстка Л. Ф. Кириленкова

Подписано в печать 01.04.10. Бумага типографская № 1.
Формат 60 × 84 $^{1}/_{16}$. Гарнитура Таймс. Ризография.
Усл. печ. л. 6,97. Уч.-изд. л. 7,27. Тираж 145 экз.
Заказ №

Учреждение образования
«Белорусский торгово-экономический
университет потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.
ЛИ № 02330/0494302 от 04.03.2009 г.

Отпечатано в учреждении образования
«Белорусский торгово-экономический
университет потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.